MPX-32 ™

Overview and System Services

Revision 3.5

Reference Manual Volume I

April 1990

323-001551-600

ENCORE

Limited Rights

This manual is supplied without representation or warranty of any kind. Encore Computer Corporation therefore assumes no responsibility and shall have no liability of any kind arising from the supply or use of this publication or any material contained herein.

Proprietary Information

The information contained herein is proprietary to Encore Computer Corporation and/or its vendors, and its use, disclosure, or duplication is subject to the restrictions stated in the standard Encore Computer Corporation License terms and conditions or the appropriate third-party sublicense agreement.

Restricted Rights

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at 252.227.7013.

Encore Computer Corporation
6901 West Sunrise Boulevard
Fort Lauderdale, Florida 33313

MPX-32 ™

Overview

Revision 3.5

Reference Manual Volume I(A)

April 1990

1 Introduction
2 Task Structure & Operation
3 Resource Management
4 Volume Resource Management
5 Resource Attachment & I/O

323-001551-600

History

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

Publication Order Number **323-001551-100**, (Revision 1, Release 3.2) was printed June, 1983.

Publication Order Number **323-001551-200**, (Revision 2, Release 3.2B) was printed March, 1985.

Publication Order Number **323-001551-201**, (Change 1 to Revision 2, Release 3.2C) was printed December, 1985.

Publication Order Number **323-001551-300**, (Revision 3, Release 3.3) was printed December, 1986.

Publication Order Number **323-001551-400**, (Revision 4, Release 3.4) was printed January, 1988.

Publication Order Number **323-001551-500**, (Revision 3.4U03) was printed October, 1989.

Publication Order Number **323-01551-600**, (Revision 3.5) was printed April, 1990.

This manual contains the following pages:

Title page
Copyright page
Title page Volume I(A)
iii/iv through xxxiii/xxxiv
1-1 through 1-24
2-1 through 2-53/2-54
3-1 through 3-37/3-38
4-1 through 4-51/4-52
5-1 through 5-65/5-66
Title page Volume I(B)
iii through xiii/xiv
6-1 through 6-253/6-254
7-1 through 7-212
A-1 through A-7/A-8
B-1 through B-42
C-1 through C-32
D-1 through D-2
E-1 through E-2
F-1 through F-2
G-1 through G-2
H-1 through H-2
I-1/I-2
J-1 through J-2
K-1 through K-2
L-1 through L-79/L-80
GL-1 through GL-10
IN-1 through IN-31/IN-32

Contents

Page

Documentation Conventions xxxi

1 Introduction

1.1 System Description 1-1
1.1.1 Hardware Interrupts/Traps 1-5
1.1.2 Software Interrupt System 1-7
1.1.3 Task Priority Levels 1-7
1.1.4 Supervision and Allocation 1-7
1.1.5 Memory Allocation 1-8
1.1.5.1 Dynamic Allocation 1-8
1.1.6 File Management 1-8
1.1.6.1 Permanent Files 1-8
1.1.6.2 Temporary Files 1-9
1.1.6.3 Random Access Files 1-9
1.1.6.4 Disk File Protection 1-9
1.1.6.5 Dedicated System Files 1-9
1.1.6.6 Multiprocessor Files 1-9
1.1.7 System Services 1-10
1.1.8 Input/Output Operations 1-10
1.1.8.1 Direct I/O 1-10
1.1.8.2 Device-Independent I/O 1-10
1.1.8.3 Logical File Codes 1-10
1.1.8.4 File Access 1-11
1.1.9 Communications Facilities 1-11
1.1.9.1 Intertask Messages 1-11
1.1.9.2 Run Requests 1-11
1.1.9.3 Global Common 1-11
1.1.9.4 Shared Images 1-11
1.1.9.5 Datapool 1-12
1.1.9.6 Internal Communications 1-12
1.1.10 Trap Processors 1-12
1.1.11 Timer Scheduler 1-12
1.1.12 Time Management 1-12
1.1.13 System Nonresident Media Mounting Task (J.MOUNT) 1-13
1.2 System Command Processors 1-13

Contents

Page

1.2.1 Terminal Services Manager (TSM) 1-13
1.2.2 Operator Communications (OPCOM) 1-14
1.2.3 Batch Processing 1-14
1.3 Program Development Utilities 1-15
1.3.1 Task Cataloging (CATALOG) 1-15
1.3.1.1 Privilege 1-15
1.3.1.2 Overlays 1-16
1.3.2 Task Debugger (AIDDB) 1-16
1.3.3 Macro Assembler (ASSEMBLE) 1-16
1.3.4 Macro Library Editor (MACLIBR) 1-16
1.3.5 Subroutine Library Editor (LIBED) 1-17
1.3.6 Datapool Editor (DPEDIT) 1-17
1.3.7 Text Editor (EDIT) 1-17
1.3.8 Volume Manager (VOLMGR) 1-17
1.3.9 Volume Formatter (J.VFMT) 1-17
1.3.10 Assembler/X32 (ASMX32) 1-17
1.3.11 Macro Librarian/X32 (MACX32) 1-17
1.3.12 Object Librarian/X32 (OBJX32) 1-18
1.3.13 Linker/X32 (LINKX32) 1-18
1.3.14 Symbolic Debugger/X32 (DEBUGX32) 1-18
1.4 Service Utilities 1-19
1.4.1 Source Update (UPDATE) 1-19
1.4.2 Media Conversion (MEDIA) 1-19
1.5 System Manager Utilities 1-20
1.5.1 M.KEY Editor (KEY) 1-20
1.5.2 MPX-32 System Start-up, Generation, and Installation (SYSGEN) 1-20
1.6 Libraries 1-21
1.6.1 Subroutine Libraries 1-21
1.6.2 System Macro Libraries 1-21
1.6.3 Other 1-21
1.7 Minimum Hardware Configuration 1-22

2 Task Structure and Operation Overview

2.1 Task Identification 2-1
2.2 Task Structure 2-2
2.2.1 Nonbase Nonshared Tasks 2-3
2.2.2 Base Nonshared Tasks 2-3
2.2.3 Multicopied Tasks 2-3

Page

2.2.4 Shared Tasks 2-3
2.2.5 Unique Tasks 2-3
2.3 Task Execution 2-7
2.3.1 Task Activation Sequencing (M.ACTV, M.PTSK) 2-7
2.3.1.1 Phase 1 of Activation 2-7
2.3.1.2 Phase 2 of Activation 2-8
2.3.2 Task Service Area (TSA) 2-8
2.4 Central Processing Unit (CPU) Scheduling 2-10
2.4.1 Execution Priorities 2-10
2.4.2 Real-Time Priority Levels (1 to 54) 2-10
2.4.3 Time-Distribution Priority Levels (55 to 64) 2-10
2.4.3.1 Priority Migration 2-11
2.4.3.2 Situational Priority Increments 2-11
2.4.3.3 Time-Quantum Controls 2-11
2.4.4 State Chain Management 2-12
2.5 Internal Processing Unit (IPU) 2-15
2.5.1 Options 2-15
2.5.2 Biased Task Prioritization 2-15
2.5.2.1 Standard CPU/IPU Scheduler 2-15
2.5.2.2 Optional CPU/IPU Scheduler 2-15
2.5.3 Nonbiased Task Prioritization 2-16
2.5.4 IPU Task Selection and Execution 2-16
2.5.5 CPU Execution of IPU Tasks 2-16
2.5.5.1 Standard CPU/IPU Scheduler 2-16
2.5.5.2 Optional CPU/IPU Scheduler 2-17
2.5.6 Priority versus Biasing 2-17
2.5.7 IPU Accounting 2-17
2.5.8 IPU Executable System Services 2-18
2.5.9 IPU Scheduling 2-18
2.6 Scheduling Task Interrupts 2-20
2.6.1 Task Interrupt Levels 2-20
2.6.1.1 Task Interrupt Receivers 2-20
2.6.1.2 Scheduling 2-20
2.6.1.3 System Service Calls from Task Interrupt Levels 2-20
2.6.1.4 Task Interrupt Context Storage 2-21
2.6.1.5 Task Interrupt Level Gating 2-21
2.6.2 User Break Interrupt Receivers (M.BRK, M.BRKXIT) 2-21

Contents

Page

2.7 Intertask Communication 2-22
2.7.1 User End-Action Receivers (M.XMEA, M.XREA, M.XIEA) 2-22
2.7.2 User Message Receivers (M.RCVR, M.GMSGP, M.XMSGR) 2-22
2.7.3 User Run Receivers (M.GRUNP, M.XRUNR) 2-22
2.7.4 Receiving Task Services 2-23
2.7.4.1 Establishing Message Receivers (M.RCVR) 2-23
2.7.4.2 Establishing Run Receivers 2-23
2.7.4.3 Executing Message Receiver Programs 2-23
2.7.4.4 Executing Run Receiver Programs 2-23
2.7.4.5 Obtaining Message Parameters (M.GMSGP) 2-24
2.7.4.6 Obtaining Run Request Parameters (M.GRUNP) 2-24
2.7.4.7 Exiting the Message Receiver (M.XMSGR) 2-24
2.7.4.8 Exiting the Run Receiver Task (M.EXIT, M.XRUNR) 2-24
2.7.4.9 Waiting for the Next Request (M.SUSP, M.ANYW, M.EAWAIT) 2-25
2.7.5 Sending Task Services 2-25
2.7.5.1 Message Send Service (M.SMSGR) 2-25
2.7.5.2 Send Run-Request Service (M.SRUNR) 2-26
2.7.5.3 Waiting for Message Completion 2-26
2.7.5.4 Waiting for Run-Request Completion 2-26
2.7.5.5 Message End-Action Processing (M.XMEA) 2-26
2.7.5.6 Run-Request End-Action Processing (M.XREA) 2-26
2.7.6 Parameter Blocks 2-26
2.7.6.1 Parameter Send Block (PSB) 2-27
2.7.6.2 Parameter Receive Block (PRB) 2-32
2.7.6.3 Receiver Exit Block (RXB) 2-33
2.7.7 User Abort Receivers (M.SUAR) 2-34
2.7.8 Task Interrupt Services Summary 2-34
2.7.9 Arithmetic Exception Handling 2-34
2.7.9.1 Establishing Exception Handler 2-37
2.7.9.2 Changing a Return Address from an Exception Handler 2-37
2.7.9.3 Exception Handler Input Arguments 2-37
2.7.9.4 Special Arithmetic Exception Processing and Ada Tasks 2-41
2.7.9.5 Exception Handler Restrictions 2-41
2.7.9.6 Related Arithmetic Exception Information 2-41

Page

2.8 CPU Dispatch Queue Area 2-42
2.9 I/O Scheduling 2-42
2.10 Swap Scheduling 2-43
2.10.1 Structure 2-43
2.10.2 Entry Conditions 2-43
2.10.2.1 Dynamic Expansion of Address Space (M.GE/M.GD, M.MEMB) 2-43
2.10.2.2 Deallocation of Memory (M.FE/M.FD, M.MEMFRE) 2-43
2.10.2.3 Request for Inswap 2-43
2.10.2.4 Change in Task Status 2-43
2.10.3 Exit Conditions 2-44
2.10.4 Selection of Inswap and Outswap Candidates 2-44
2.10.4.1 Outswap Process 2-45
2.10.4.2 Inswap Process 2-45
2.11 Task Termination Sequencing 2-46
2.11.1 Nonbase Mode Exit Task (M.EXIT) 2-46
2.11.2 Abort Task (M.BORT) 2-46
2.11.3 Delete Task (M.DELTSK) 2-46
2.11.4 Base Mode Exit Task (M_EXIT) 2-46
2.12 Task-Synchronized Access to Common Resources 2-49
2.13 MPX-32 Faults/Traps and Miscellaneous Interrupts 2-51
2.14 Real-Time Task Accounting On/Off 2-53

3 Resource Management Overview

3.1 General Resource Management 3-1
3.2 Support for Resource Types 3-1
3.2.1 Physical Resources 3-1
3.2.2 Logical Resources 3-1
3.3 Support for Resource Functions 3-2
3.3.1 Resource Creation 3-2
3.3.2 Resource Deletion 3-2
3.3.3 Resource Attachment 3-2
3.3.3.1 Static Allocation 3-3
3.3.3.2 Dynamic Allocation 3-3
3.3.4 Resource Access 3-3
3.3.4.1 Device Level 3-3
3.3.4.2 Execute Channel Program Level 3-3

Contents

Page

3.3.4.3 Logical Device Level 3-4
3.3.4.4 Logical File Level 3-4
3.3.4.5 Blocked Level 3-4
3.3.5 Resource Detachment 3-4
3.3.6 Resource Inquiry 3-5
3.3.6.1 Inquiry of Unattached Resources 3-5
3.3.6.2 Inquiry of Attached Resources 3-5
3.3.7 Resource Attribute Modification 3-5
3.4 Resource Attributes 3-6
3.4.1 Protection 3-6
3.4.2 Shareable Resources 3-6
3.4.2.1 Exclusive Use 3-7
3.4.2.2 Explicit Use 3-7
3.4.2.3 Implicit Use 3-8
3.5 Resource Access Attributes 3-8
3.5.1 Access Attributes for Volumes 3-8
3.5.2 Access Attributes for Directories 3-9
3.5.3 Access Attributes for Files 3-10
3.5.4 Access Attributes for Memory Partitions 3-10
3.6 Management Attributes 3-11
3.6.1 Extension Attribute 3-11
3.6.1.1 Manual Extension Attribute 3-11
3.6.1.2 Automatic Extension Attribute 3-11
3.6.2 Contiguity Attribute 3-11
3.6.3 Maximum and Minimum Extension Attributes 3-12
3.6.4 Maximum File Size Attribute 3-12
3.6.5 Shared Attribute 3-12
3.6.6 End-Of-File Management Attribute 3-12
3.6.7 Fast Access Attribute 3-13
3.6.8 Zero Attribute 3-13
3.6.9 File Type Attribute 3-14
3.6.10 No-Save Attribute 3-14
3.7 Operating System Memory Allocation 3-15
3.7.1 I/O Buffer and I/O Queues 3-15
3.7.2 Blocking Buffers for Blocked I/O 3-15
3.7.3 Large Buffers for Blocked Files 3-15
3.8 Memory Classes 3-16
3.9 Memory Allocation for Tasks 3-16

Page

3.9.1 Demand Page Processing (CONCEPT 32/2000 Only) 3-17
3.9.2 Static Memory Allocation 3-18
3.9.3 Dynamic Address Space Expansion/Contraction (M.GE, M.FE, M.GD, M.FD, M.MEMB, M.MEMFRE) 3-19
3.9.4 Extended Indexed Data Space for Nonbase Mode Tasks 3-20
3.9.5 Intertask Shared Global Memory and Datapool Memory (M.INCLUDE, M.EXCLUDE) 3-20
3.9.6 Shared Procedures for Nonbase Mode Tasks 3-21
3.9.7 Multiprocessor Shared Memory 3-22
3.10 Extended MPX-32 (Expanded Execution Space) 3-23
3.10.1 SYSGEN Information for Extended MPX-32 3-27
3.10.2 SYSGEN EXTDMPX Directive 3-28
3.10.3 SYSGEN Aborts and Errors for Extended MPX-32 3-29
3.10.4 How to Create an Extended MPX-32 System 3-29
3.10.5 How to Relocate Extended MPX-32 3-31
3.10.6 CATALOG EXTDMPX Directive 3-32
3.10.7 TSM EXTDMPX Directive 3-33
3.11 Extended TSA (Expanded Execution Space) 3-34
3.11.1 Relocating the TSA 3-35
3.12 Mapped Out Option (CONCEPT 32/2000 Only) 3-37

4 Volume Resource Management

4.1 Symbolic Resource Management 4-1
4.1.1 Types of Resources 4-1
4.1.2 Classes of Resources 4-2
4.1.3 Classes of Resource Users 4-2
4.1.4 Shareable Resource Control Mechanisms 4-3
4.2 General Resource Control 4-4
4.2.1 Enqueue and Synchronous Notification Mechanism 4-4
4.2.2 Dequeue Mechanism 4-4
4.3 Shareable Resource Access Control 4-5
4.3.1 Shareable Resource Locking 4-5
4.3.2 Shareable Resource Synchronization 4-5
4.4 Standard Disk Structure 4-6
4.4.1 Directory Structure 4-6

Contents

Page

4.4.2 Root Directory 4-6
4.4.3 Current Working Directory 4-6
4.5 Pathnames 4-7
4.5.1 Executing Pathnames 4-7
4.5.2 Fully Qualified Pathnames 4-8
4.5.3 Partially Qualified Pathnames 4-9
4.5.4 Fully Qualified Pathnames for Directories Only 4-10
4.5.5 Partially Qualified Directory Pathnames 4-11
4.6 Resource Protection 4-13
4.7 System Administration 4-13
4.8 Volumes 4-14
4.8.1 Overview of Formatted Volumes 4-14
4.8.2 Formatted Volume Type 4-15
4.8.2.1 System Volume 4-15
4.8.2.2 User Volume 4-15
4.8.2.3 Multiprocessor Volume 4-16
4.8.3 Access Attributes for Formatted Volumes 4-16
4.8.3.1 Public Attribute 4-16
4.8.3.2 Nonpublic Attribute 4-16
4.8.4 Mounting Formatted Volumes 4-17
4.8.4.1 Physical Mount 4-17
4.8.4.2 Logical Mount 4-18
4.8.5 Dismounting Formatted Volumes 4-18
4.8.5.1 Logical Dismount 4-18
4.8.5.2 Physical Dismount 4-19
4.8.6 Automatic Mounting at System Boot 4-20
4.8.7 Components of a Volume 4-20
4.8.7.1 Boot Block 4-22
4.8.7.2 Volume Descriptor 4-22
4.8.7.3 Resource Descriptors (RDs) 4-22
4.9 Directories 4-24
4.9.1 Volume Root Directory 4-26
4.9.2 Creating Directories 4-26
4.9.3 Protecting Directories 4-28
4.9.4 Protecting Directory Entries 4-28
4.9.5 Using Directories 4-28

Page

4.10 Files 4-29
4.10.1 File Attributes 4-29
4.10.2 Obtaining File Space 4-30
4.10.2.1 Granularity 4-30
4.10.2.2 Contiguity 4-30
4.10.2.3 Extendibility 4-30
4.10.2.4 Size 4-31
4.10.3 File Names and Fast Access 4-31
4.10.4 File Protection 4-32
4.10.5 Permanent Files 4-32
4.10.6 Creating Files 4-32
4.10.7 Attaching Files 4-33
4.10.8 Assigning Files 4-33
4.10.9 Opening Files 4-34
4.10.10 File Operations 4-34
4.10.10.1 Sequential Access 4-35
4.10.10.2 Random Access 4-35
4.10.11 File Positioning 4-35
4.10.11.1 Absolute File Positioning Operations 4-36
4.10.11.2 Relative File Positioning Operations 4-36
4.10.12 File Access Modes 4-36
4.10.12.1 Read Mode 4-37
4.10.12.2 Write Mode 4-38
4.10.12.3 Modify Mode 4-38
4.10.12.4 Update Mode 4-39
4.10.12.5 Append Mode 4-39
4.10.13 Sharing Files 4-40
4.10.14 Closing Files 4-40
4.10.15 Detaching Files 4-41
4.10.16 Deleting Files 4-41
4.10.17 Temporary Files 4-41
4.10.17.1 Creating Temporary Files 4-41
4.10.17.2 Assigning Temporary Files 4-41
4.10.17.3 Opening and Accessing Temporary Files 4-42
4.10.17.4 Deleting and Detaching Temporary Files 4-42
4.10.17.5 Making Temporary Files Permanent 4-42

Page

4.11 Memory Partitions - Nonbase Mode of Addressing 4-43
4.11.1 Creating Memory Partitions 4-43
4.11.2 Protecting Memory Partitions 4-43
4.11.3 Attaching Memory Partitions 4-44
4.11.4 Accessing Memory Partitions 4-44
4.11.5 Detaching Memory Partitions 4-44
4.11.6 Deleting Memory Partitions 4-44
4.11.7 Sharing Memory Partitions 4-45
4.12 Shared Images 4-45
4.12.1 Created Shared Images 4-45
4.12.2 Protecting Shared Images 4-45
4.12.3 Attaching Shared Images 4-46
4.12.4 Accessing Shared Images 4-46
4.12.5 Detaching Shared Images 4-46
4.13 Multiprocessor Shared Volumes 4-47
4.13.1 Multiprocessor Resources 4-47
4.13.2 Multiprocessor Resource Access 4-47
4.13.3 Mounting Multiprocessor Volumes 4-49
4.13.4 Multiprocessor Resource Restrictions 4-49
4.13.4.1 EOF Management 4-49
4.13.4.2 EOM Management 4-50
4.13.4.3 Resource Deadlocks 4-50
4.13.4.4 Reserve/Release Multiported Disk Services (M.RESP/M.RELP) 4-50
4.13.5 Optimum Use of Multiprocessor Resources 4-51

5 Resource Assignment/Allocation and I/O

5.1 Introduction 5-1
5.2 MPX-32 Logical I/O (Device-Independent) 5-1
5.2.1 Logical File Codes 5-2
5.2.2 File Control Blocks 5-2
5.2.2.1 Logical I/O Initiation 5-2
5.2.3 Assignment vs. Allocation 5-2
5.2.4 Logical File Code Assignment 5-4
5.2.4.1 Making Assignments via Resource Requirement Summary (RRS) 5-4
5.2.4.2 Temporary File Assignments 5-14
5.2.5 Opening a Resource for Logical I/O 5-14

Page

5.3 Resource Conflicts and Error Handling 5-15
5.3.1 Status Posting and Return Conventions 5-16
5.4 MPX-32 Volume Resource Access 5-20
5.4.1 Volume Resource Space Management 5-20
5.4.2 Temporary vs. Permanent Files 5-20
5.4.3 System Directory 5-21
5.5 MPX-32 Device Access 5-21
5.5.1 Magnetic Tape 5-21
5.5.2 Unformatted Media 5-25
5.5.3 Examples of Device Identification Levels 5-28
5.5.4 GPMC Devices 5-28
5.5.5 NULL Device 5-28
5.5.6 System Console 5-28
5.5.7 Special File Attributes 5-29
5.6 Samples 5-30
5.7 Device-Independent I/O Processing 5-32
5.7.1 Wait I/O 5-32
5.7.1.1 Wait I/O Errors 5-32
5.7.1.2 Wait I/O Exit and I/O Abort Processing 5-33
5.7.1.3 Error Processing and Status Inhibit 5-33
5.7.2 No-Wait I/O 5-33
5.7.2.1 No-Wait I/O Complete Without Errors 5-33
5.7.2.2 No-Wait I/O Complete with Errors 5-34
5.7.2.3 No-Wait End-Action Return to IOCS 5-34
5.7.3 Direct I/O 5-34
5.7.4 Blocked I/O 5-34
5.7.5 End-of-File and End-of-Medium Processing 5-36
5.7.6 Software End-of-File for Unblocked Files 5-36
5.8 Spooled Output with Print or Punch Attribute 5-38
5.9 Setting Up File Control Blocks for Device-Independent I/O 5-38
5.9.1 Macros (M.DFCB/M.DFCBE) 5-39
5.9.2 Sample FCB Set-up Nonmacro 5-40
5.9.3 Sample FCB Set-up Macro 5-40

Page

5.10 Setting Up TCPBs for the System Console 5-41
5.11 MPX-32 Device-Dependent I/O 5-43
5.11.1 Device-Dependent I/O Processing Overview 5-43
5.11.2 Operational Description of Execute Channel Program (EXCPM) 5-44
5.11.2.1 Logical Channel Program 5-44
5.11.2.2 Physical Channel Program 5-44
5.11.2.3 Post Program-Controlled Interrupt (PPCI) End-Action Receiver 5-44
5.11.2.4 Restrictions 5-45
5.11.2.5 Setting Up File Control Blocks for EXCPM Requests 5-46
5.11.2.6 Post Program-Controlled Interrupt Notification Packet 5-47
5.11.2.7 Macros (M.FCBEXP) 5-48
5.12 Resident Executive Services (H.REXS) 5-49
5.13 Resource Management (H.REMM) 5-50
5.14 Volume Management Module (H.VOMM) 5-51
5.14.1 H.VOMM Conventions 5-51
5.14.1.1 Entry Point Conventions 5-51
5.14.1.2 Pathnames 5-51
5.14.1.3 Pathname Blocks (PNB) 5-52
5.14.1.4 Resource Identifiers 5-55
5.14.1.5 Allocation Units 5-55
5.14.1.6 File Segment Definitions 5-55
5.14.2 Calling/Return Parameter Conventions 5-56
5.14.2.1 Unused Register 5-56
5.14.2.2 Specifying a Volume Resource 5-56
5.14.2.3 Status Codes 5-57
5.14.2.4 Caller Notification Packet (CNP) 5-58
5.14.2.5 Pathnames/Pathname Blocks 5-59
5.14.2.6 Resource Create Block (RCB) 5-59
5.14.3 Bad Block Handling 5-64
5.14.4 Services 5-65

Page

6 Nonbase Mode System Services

6.1 Overview 6-1
6.1.1 Syntax Rules and Descriptions 6-2
6.1.2 IPU Executable Nonbase Mode System Services 6-3
6.2 Macro-Callable System Services 6-4
6.2.1 M.ACTV - Activate Task 6-5
6.2.2 M.ADRS - Memory Address Inquiry 6-6
6.2.3 M.ANYW - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt 6-7
6.2.4 M.ASSN - Assign and Allocate Resource 6-8
6.2.5 M.ASYNCH - Set Asynchronous Task Interrupt 6-10
6.2.6 M.BACK - Backspace Record or File 6-11
6.2.7 M.BATCH - Batch Job Entry 6-13
6.2.8 M.BBTIM - Acquire Current Date/Time in Byte Binary Format 6-15
6.2.9 M.BORT - Abort Specified Task, Abort Self, or Abort with Extended Message 6-16
6.2.9.1 M.BORT - Specified Task 6-16
6.2.9.2 M.BORT - Self 6-17
6.2.9.3 M.BORT - With Extended Message 6-18
6.2.10 M.BRK - Break/Task Interrupt Link/Unlink 6-19
6.2.11 M.BRKXIT - Exit from Task Interrupt Level 6-19
6.2.12 M.BTIM - Acquire Current Date/Time in Binary Format 6-20
6.2.13 M.CLOSER - Close Resource 6-21
6.2.14 M.CLSE - Close File 6-23
6.2.15 M.CMD - Get Command Line 6-24
6.2.16 M.CONABB - Convert ASCII Date/Time to Byte Binary Format 6-25
6.2.17 M.CONADB - Convert ASCII Decimal to Binary 6-26
6.2.18 M.CONAHB - Convert ASCII Hexadecimal to Binary 6-27
6.2.19 M.CONASB - Convert ASCII Date/Time to Standard Binary 6-28
6.2.20 M.CONBAD - Convert Binary to ASCII Decimal 6-29
6.2.21 M.CONBAF - Convert Binary Date/Time to ASCII Format 6-30
6.2.22 M.CONBAH - Convert Binary to ASCII Hexadecimal 6-31
6.2.23 M.CONBBA - Convert Byte Binary Date/Time to ASCII 6-32

Contents

Page

6.2.24 M.CONBBY - Convert Binary Date/Time to Byte Binary 6-33
6.2.25 M.CONBYB - Convert Byte Binary Date/Time to Binary 6-34
6.2.26 M.CONN - Connect Task to Interrupt 6-35
6.2.27 M.CPERM - Create Permanent File.....6-37
6.2.28 M.CTIM - Convert System Date/Time Format 6-39
6.2.29 M.CWAT - System Console Wait 6-41
6.2.30 M.DASN - Deassign and Deallocate Resource.....6-42
6.2.31 M.DATE - Date and Time Inquiry.....6-44
6.2.32 M.DEBUG - Load and Execute Interactive Debugger 6-45
6.2.33 M.DEFT - Change Defaults 6-46
6.2.34 M.DELR - Delete Resource 6-48
6.2.35 M.DELTSK - Delete Task.....6-50
6.2.36 M.DEVID - Get Device Mnemonic or Type Code.....6-52
6.2.37 M.DIR - Create Directory.....6-53
6.2.38 M.DISCON - Disconnect Task from Interrupt 6-55
6.2.39 M.DLTT - Delete Timer Entry.....6-56
6.2.40 M.DMOUNT - Dismount Volume 6-57
6.2.41 M.DSMI - Disable Message Task Interrupt.....6-59
6.2.42 M.DSUB - Disable User Break Interrupt.....6-60
6.2.43 M.DUMP - Memory Dump Request.....6-61
6.2.44 M.EAWAIT - End Action Wait 6-63
6.2.45 M.ENMI - Enable Message Task Interrupt.....6-64
6.2.46 M.ENUB - Enable User Break Interrupt.....6-65
6.2.47 M.ENVRMT - Get Task Environment.....6-66
6.2.48 M.EXCLUDE - Exclude Memory Partition.....6-67
6.2.49 M.EXIT - Terminate Task Execution 6-69
6.2.50 M.EXTD - Extend File 6-70
6.2.51 M.FD - Free Dynamic Extended Indexed Data Space 6-72
6.2.52 M.FE - Free Dynamic Task Execution Space.....6-73
6.2.53 M.FWRD - Advance Record or File 6-74
6.2.54 M.GADRL - Get Address Limits 6-76
6.2.55 M.GADRL2 - Get Address Limits 6-77
6.2.56 M.GD - Get Dynamic Extended Data Space 6-78
6.2.57 M.GDD - Get Dynamic Extended Discontiguous Data Space.....6-79
6.2.58 M.GE - Get Dynamic Task Execution Space 6-80
6.2.59 M.GETDEF - Get Definition for Terminal Function.....6-81
6.2.60 M.GMSGP - Get Message Parameters.....6-83
6.2.61 M.GRUNP - Get Run Parameters 6-84

Page

6.2.62 M.GTIM - Acquire System Date/Time in Any Format 6-85
6.2.63 M.GTSAD - Get TSA Start Address 6-86
6.2.64 M.HOLD - Program Hold Request 6-87
6.2.65 M.ID - Get Task Number 6-88
6.2.66 M.INCLUDE - Include Memory Partition 6-90
6.2.67 M.INQUIRY - Resource Inquiry 6-93
6.2.68 M.INT - Activate Task Interrupt 6-97
6.2.69 M.IPUBS - Set IPU Bias 6-98
6.2.70 M.LOC - Read Descriptor 6-99
6.2.71 M.LOCK - Set Exclusive Resource Lock 6-101
6.2.72 M.LOGR - Log Resource or Directory 6-103
6.2.72.1 Resource Specifications for Pathnames 6-103
6.2.72.2 Resource Specifications for Pathname Blocks 6-104
6.2.72.3 Resource Specifications for a Resource Identifier ... 6-104
6.2.72.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index 6-104
6.2.73 M.MEM - Create Memory Partition 6-108
6.2.74 M.MEMB - Get Memory in Byte Increments 6-110
6.2.75 M.MEMFRE - Free Memory in Byte Increments 6-111
6.2.76 M.MOD - Modify Descriptor 6-112
6.2.77 M.MODU - Modify Descriptor User Area 6-114
6.2.78 M.MOUNT - Mount Volume 6-115
6.2.79 M.MOVE - Move Data to User Address 6-117
6.2.80 M.MYID - Get Task Number 6-118
6.2.81 M.NEWRRS - Reformat RRS Entry 6-119
6.2.82 M.OLAY - Load Overlay Segment 6-121
6.2.83 M.OPENR - Open Resource 6-122
6.2.84 M.OSREAD - Physical Memory Read 6-124
6.2.85 M.OSWRIT - Physical Memory Write 6-125
6.2.86 M.PGOD - Task Option Doubleword Inquiry 6-126
6.2.87 M.PGOW - Task Option Word Inquiry 6-127
6.2.88 M.PNAM - Reconstruct Pathname 6-128
6.2.89 M.PNAMB - Convert Pathname to Pathname Block 6-129
6.2.90 M.PRIL - Change Priority Level 6-131
6.2.91 M.PRIV - Reinstate Privilege Mode to Privilege Task 6-132
6.2.92 M.PTSK - Parameter Task Activation 6-133
6.2.93 M.QATIM - Acquire Current Date/Time in ASCII Format ... 6-138
6.2.94 M.RADDR - Get Real Physical Address 6-139
6.2.95 M.RCVR - Receive Message Link Address 6-140

Page

6.2.96 M.READ - Read Record 6-141
6.2.97 M.RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode 6-142
6.2.98 M.RENAM - Rename File 6-143
6.2.99 M.REPLAC - Replace Permanent File 6-144
6.2.100 M.RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode 6-145
6.2.101 M.REWRIT - Rewrite Descriptor 6-146
6.2.102 M.REWRTU - Rewrite Descriptor User Area 6-147
6.2.103 M.ROPL - Reset Option Lower 6-148
6.2.104 M.RRES - Release Channel Reservation 6-149
6.2.105 M.RSML - Resourcemark Lock 6-150
6.2.106 M.RSMU - Resourcemark Unlock 6-152
6.2.107 M.RSRV - Reserve Channel 6-153
6.2.108 M.RWND - Rewind File 6-154
6.2.109 M.SETS - Set User Status Word 6-155
6.2.110 M.SETSYNC - Set Synchronous Resource Lock 6-157
6.2.111 M.SETT - Create Timer Entry 6-159
6.2.112 M.SMSGR - Send Message to Specified Task 6-162
6.2.113 M.SOPL - Set Option Lower 6-163
6.2.114 M.SRUNR - Send Run Request to Specified Task 6-164
6.2.115 M.SUAR - Set User Abort Receiver Address 6-166
6.2.116 M.SUME - Resume Task Execution 6-167
6.2.117 M.SURE - Suspend/Resume 6-168
6.2.118 M.SUSP - Suspend Task Execution 6-169
6.2.119 M.SYNCH - Set Synchronous Task Interrupt 6-170
6.2.120 M.TBRKON - Trap Online User's Task 6-171
6.2.121 M.TDAY - Time-of-Day Inquiry 6-172
6.2.122 M.TEMP - Create Temporary File 6-173
6.2.123 M.TEMPER - Change Temporary File to Permanent File 6-175
6.2.124 M.TRNC - Truncate File 6-177
6.2.125 M.TSCAN - Scan Terminal Input Buffer 6-178
6.2.126 M.TSMPC - TSM Procedure Call 6-179
6.2.127 M.TSTE - Arithmetic Exception Inquiry 6-182
6.2.128 M.TSTS - Test User Status Word 6-183
6.2.129 M.TSTT - Test Timer Entry 6-184
6.2.130 M.TURNON - Activate Program at Given Time-of-Day 6-185
6.2.131 M.TYPE - System Console Type 6-187
6.2.132 M.UNLOCK - Release Exclusive Resource Lock 6-188
6.2.133 M.UNSYNC - Release Synchronous Resource Lock 6-190

Page

6.2.134 M.UPRIV - Change Task to Unprivileged Mode 6-192
6.2.135 M.UPSP - Upspace 6-193
6.2.136 M.VADDR - Validate Address Range 6-194
6.2.137 M.WAIT - Wait I/O 6-195
6.2.138 M.WEOF - Write EOF 6-196
6.2.139 M.WRIT - Write Record 6-197
6.2.140 M.XBRKR - Exit from Task Interrupt Level 6-198
6.2.141 M.XIEA - No-Wait I/O End-Action Return 6-199
6.2.142 M.XMEA - Exit from Message End-Action Routine 6-200
6.2.143 M.XMSGR - Exit from Message Receiver 6-201
6.2.144 M.XREA - Exit from Run Request End-Action Routine 6-202
6.2.145 M.XRUNR - Exit Run Receiver 6-203
6.2.146 M.XTIME - Task CPU Execution Time 6-204
6.3 Nonmacro-Callable System Services 6-205
6.3.1 Allocate File Space 6-206
6.3.1.1 Clean-up Mode 6-206
6.3.1.2 Normal Mode 6-206
6.3.2 Allocate Resource Descriptor 6-208
6.3.3 Create Temporary File 6-209
6.3.3.1 VOMM Internal Call 6-209
6.3.3.2 External Call 6-209
6.3.3.3 Default File Attributes 6-209
6.3.3.4 Volume Selection 6-209
6.3.4 Deallocate File Space 6-211
6.3.5 Deallocate Resource Descriptor 6-212
6.3.6 Debug Link Service 6-213
6.3.7 Eject/Purge Routine 6-214
6.3.8 Erase or Punch Trailer 6-215
6.3.9 Execute Channel Program 6-216
6.3.10 Get Extended Memory Array 6-217
6.3.11 Read/Write Authorization File 6-218
6.3.12 Release FHD Port 6-219
6.3.13 Reserve FHD Port 6-220
6.4 Compatible System Services 6-221
6.4.1 M.ALOC - Allocate File or Peripheral Device 6-222
6.4.2 M.CDJS - Submit Job from Disk File 6-226
6.4.3 M.CREATE - Create Permanent File 6-228
6.4.4 M.DALC - Deallocate File or Peripheral Device 6-231

Page

6.4.5 M.DELETE - Delete Permanent File or Non-SYSGEN Memory Partition 6-232
6.4.6 M.EXCL - Free Shared Memory 6-233
6.4.7 M.FADD - Permanent File Address Inquiry 6-234
6.4.8 M.FILE - Open File 6-236
6.4.9 M.FSLR - Release Synchronization File Lock 6-237
6.4.10 M.FSLS - Set Synchronization File Lock 6-238
6.4.11 M.FXLR - Release Exclusive File Lock 6-240
6.4.12 M.FXLS - Set Exclusive File Lock 6-241
6.4.13 M.INCL - Get Shared Memory 6-242
6.4.14 M.LOG - Permanent File Log 6-244
6.4.15 M.PDEV - Physical Device Inquiry 6-246
6.4.16 M.PERM - Change Temporary File to Permanent 6-248
6.4.17 M.SHARE - Share Memory with Another Task 6-250
6.4.18 M.SMULK - Unlock and Dequeue Shared Memory 6-252
6.4.19 M.USER - User Name Specification 6-253

7 Base Mode System Services

7.1 General Description 7-1
7.1.1 Syntax Rules and Descriptions 7-2
7.1.1.1 Parameter Specification 7-2
7.1.2 IPU Executable Base Mode System Services 7-5
7.2 Macro-Callable System Services 7-6
7.2.1 M_ACTV - Activate Task 7-7
7.2.2 M_ADRS - Memory Address Inquiry 7-8
7.2.3 M_ADVANCE - Advance Record or File 7-9
7.2.4 M_ANYWAIT - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt 7-11
7.2.5 M_ASSIGN - Assign and Allocate Resource 7-12
7.2.6 M_ASYNCH - Set Asynchronous Task Interrupt 7-14
7.2.7 M_AWAITACTION - End Action Wait 7-15
7.2.8 M_BACKSPACE - Backspace Record or File 7-16
7.2.9 M_BATCH - Batch Job Entry 7-18
7.2.10 M_BBTIM - Acquire Current Date/Time in Byte Binary Format 7-20
7.2.11 M_BORT - Abort Specified Task, Abort Self, or Abort with Extended Message 7-21
7.2.11.1 M_BORT - Abort Specified Task 7-21
7.2.11.2 M_BORT - Abort Self 7-22

Page

7.2.11.3 M_BORT - Abort with Extended Message 7-23
7.2.12 M_BRK - Break/Task Interrupt Link/Unlink 7-24
7.2.13 M_BRKXIT - Exit from Task Interrupt Level 7-24
7.2.14 M_BTIM - Acquire Current Date/Time in Binary Format 7-25
7.2.15 M_CHANPROGFCB - Execute Channel Program File Control Block 7-26
7.2.16 M_CLOSER - Close Resource 7-27
7.2.17 M_CLSE - Close File 7-29
7.2.18 M_CMD - Get Command Line 7-30
7.2.19 M_CONABB - Convert ASCII Date/Time to Byte Binary Format 7-31
7.2.20 M_CONADB - Convert ASCII Decimal to Binary 7-32
7.2.21 M_CONAHB - Convert ASCII Hexadecimal to Binary 7-33
7.2.22 M_CONASB - Convert ASCII Date/Time to Standard Binary 7-34
7.2.23 M_CONBAD - Convert Binary to ASCII Decimal 7-35
7.2.24 M_CONBAF - Convert Binary Date/Time to ASCII Format 7-36
7.2.25 M_CONBAH - Convert Binary to ASCII Hexadecimal 7-37
7.2.26 M_CONBBA - Convert Byte Binary Date/Time to ASCII 7-38
7.2.27 M_CONBBY - Convert Binary Date/Time to Byte Binary 7-39
7.2.28 M_CONBYB - Convert Byte Binary Date/Time to Binary 7-40
7.2.29 M_CONN - Connect Task to Interrupt 7-41
7.2.30 M_CONSTRUCTPATH - Reconstruct Pathname 7-42
7.2.31 M_CONVERTTIME - Convert Time 7-43
7.2.32 M_CREATEFCB - Create File Control Block 7-45
7.2.33 M_CREATEP - Create Permanent File 7-46
7.2.34 M_CREATET - Create Temporary File 7-48
7.2.35 M_CTIM - Convert System Date/Time Format 7-50
7.2.36 M_CWAT - System Console Wait 7-51
7.2.37 M_DATE - Date and Time Inquiry 7-52
7.2.38 M_DEASSIGN - Deassign and Deallocate Resource 7-53
7.2.39 M_DEBUG - Load and Execute Interactive Debugger 7-55
7.2.40 M_DEFT - Change Defaults 7-56
7.2.41 M_DELETER - Delete Resource 7-57
7.2.42 M_DELTSK - Delete Task 7-59
7.2.43 M_DEVID - Get Device Mnemonic or Type Code 7-60
7.2.44 M_DIR - Create Directory 7-61
7.2.45 M_DISCON - Disconnect Task from Interrupt 7-63
7.2.46 M_DISMOUNT - Dismount Volume 7-64
7.2.47 M_DLTT - Delete Timer Entry 7-66

Contents

Page

7.2.48 M_DSMI - Disable Message Task Interrupt 7-67
7.2.49 M_DSUB - Disable User Break Interrupt 7-68
7.2.50 M_DUMP - Memory Dump Request 7-69
7.2.51 M_ENMI - Enable Message Task Interrupt 7-70
7.2.52 M_ENUB - Enable User Break Interrupt 7-71
7.2.53 M_ENVRMT - Get Task Environment 7-72
7.2.54 M_EXCLUDE - Exclude Shared Image 7-73
7.2.55 M_EXIT - Terminate Task Execution 7-75
7.2.56 M_EXTENDFILE - Extend File 7-76
7.2.57 M_EXTSTS - Exit With Status 7-78
7.2.58 M_FREEMEMBYTES - Free Memory in Byte Increments 7-79
7.2.59 M_GETCTX - Get User Context 7-80
7.2.60 M_GETDEF - Get Definition for Terminal Function 7-81
7.2.61 M_GETMEMBYTES - Get Memory in Byte Increments 7-83
7.2.62 M_GETTIME - Get Current Date and Time 7-84
7.2.63 M_GMSGP - Get Message Parameters 7-86
7.2.64 M_GRUNP - Get Run Parameters 7-87
7.2.65 M_GTIM - Acquire System Date/Time in Any Format 7-88
7.2.66 M_GTSAD - Get TSA Start Address 7-89
7.2.67 M_HOLD - Program Hold Request 7-90
7.2.68 M_ID - Get Task Number 7-91
7.2.69 M_INCLUDE - Include Shared Image 7-93
7.2.70 M_INQUIRER - Resource Inquiry 7-96
7.2.71 M_INT - Activate Task Interrupt 7-101
7.2.72 M_IPUBS - Set IPU Bias 7-102
7.2.73 M_LIMITS - Get Base Mode Task Address Limits 7-103
7.2.74 M_LOCK - Set Exclusive Resource Lock 7-104
7.2.75 M_LOGR - Log Resource or Directory 7-106
7.2.75.1 Resource Specifications for Pathnames 7-106
7.2.75.2 Resource Specifications for Pathname Blocks 7-107
7.2.75.3 Resource Specifications for a Resource Identifier 7-107
7.2.75.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index 7-107
7.2.76 M_MEM - Create Memory Partition 7-111
7.2.77 M_MOD - Modify Descriptor 7-113
7.2.78 M_MODU - Modify Descriptor User Area 7-115
7.2.79 M_MOUNT - Mount Volume 7-116
7.2.80 M_MOVE - Move Data to User Address 7-118
7.2.81 M_MYID - Get Task Number 7-120
7.2.82 M_OPENR - Open Resource 7-121

Page

7.2.83 M_OPTIONDWORD - Task Option Doubleword Inquiry 7-124
7.2.84 M_OPTIONWORD - Task Option Word Inquiry 7-125
7.2.85 M_OSREAD - Physical Memory Read 7-126
7.2.86 M_OSWRIT - Physical Memory Write 7-127
7.2.87 M_PNAMB - Convert Pathname to Pathname Block 7-129
7.2.88 M_PRIL - Change Priority Level 7-131
7.2.89 M_PRIVMODE - Reinstate Privilege Mode to Privilege Task 7-132
7.2.90 M_PTSK - Parameter Task Activation 7-133
7.2.91 M_PUTCTX - Put User Context 7-138
7.2.92 M_QATIM - Acquire Current Date/Time in ASCII Format ..7-139
7.2.93 M_RADDR - Get Real Physical Address 7-140
7.2.94 M_RCVR - Receive Message Link Address 7-141
7.2.95 M_READ - Read Record 7-142
7.2.96 M_READD - Read Descriptor 7-144
7.2.97 M_RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode 7-145
7.2.98 M_RENAME - Rename File 7-146
7.2.99 M_REPLACE - Replace Permanent File 7-147
7.2.100 M_RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode 7-148
7.2.101 M_REWIND - Rewind File 7-149
7.2.102 M_REWRIT - Rewrite Descriptor 7-150
7.2.103 M_REWRTU - Rewrite Descriptor User Area 7-151
7.2.104 M_ROPL - Reset Option Lower 7-152
7.2.105 M_RRES - Release Channel Reservation 7-153
7.2.106 M_RSML - Resourcemark Lock 7-154
7.2.107 M_RSMU - Resourcemark Unlock 7-155
7.2.108 M_RSRV - Reserve Channel 7-156
7.2.109 M_SETERA - Set Exception Return Address 7-157
7.2.110 M_SETEXA - Set Exception Handler 7-158
7.2.111 M_SETS - Set User Status Word 7-159
7.2.112 M_SETSYNC - Set Synchronous Resource Lock 7-161
7.2.113 M_SETT - Create Timer Entry 7-163
7.2.114 M_SMSGR - Send Message to Specified Task 7-166
7.2.115 M_SOPL - Set Option Lower 7-167
7.2.116 M_SRUNR - Send Run Request to Specified Task 7-168
7.2.117 M_SUAR - Set User Abort Receiver Address 7-170
7.2.118 M_SUME - Resume Task Execution 7-171

Contents

Page

7.2.119 M_SURE - Suspend/Resume 7-172
7.2.120 M_SUSP - Suspend Task Execution 7-173
7.2.121 M_SYNCH - Set Synchronous Task Interrupt 7-174
7.2.122 M_TBRKON - Trap Online User's Task 7-175
7.2.123 M_TDAY - Time-of-Day Inquiry 7-176
7.2.124 M_TEMPFILETOPERM - Change Temporary File to Permanent File 7-177
7.2.125 M_TRUNCATE - Truncate File 7-179
7.2.126 M_TSCAN - Scan Terminal Input Buffer 7-180
7.2.127 M_TSMPC - TSM Procedure Call 7-181
7.2.128 M_TSTE - Arithmetic Exception Inquiry 7-184
7.2.129 M_TSTS - Test User Status Word 7-185
7.2.130 M_TSTT - Test Timer Entry 7-186
7.2.131 M_TURNON - Activate Program at Given Time-of-Day 7-200
7.2.132 M_TYPE - System Console Type 7-189
7.2.133 M_UNLOCK - Release Exclusive Resource Lock 7-190
7.2.134 M_UNPRIVMODE - Change Task to Unprivileged Mode 7-192
7.2.135 M_UNSYNC - Release Synchronous Resource Lock 7-193
7.2.136 M_UPSP - Upspace 7-195
7.2.137 M_VADDR - Validate Address Range 7-196
7.2.138 M_WAIT - Wait I/O 7-197
7.2.139 M_WRITE - Write Record 7-198
7.2.140 M_WRITEEOF - Write EOF 7-199
7.2.141 M_XBRKR - Exit from Task Interrupt Level 7-200
7.2.142 M_XIEA - No-Wait I/O End-Action Return 7-201
7.2.143 M_XMEA - Exit from Message End-Action Routine 7-202
7.2.144 M_XMSGR - Exit from Message Receiver 7-203
7.2.145 M_XREA - Exit from Run Request End-Action Routine 7-204
7.2.146 M_XRUNR - Exit Run Receiver 7-205
7.2.147 M_XTIME - Task CPU Execution Time 7-206
7.3 Nonmacro-Callable System Services 7-207
7.3.1 Debug Link Service 7-207
7.3.2 Eject/Purge Routine 7-208
7.3.3 Erase or Punch Trailer 7-209
7.3.4 Execute Channel Program 7-210
7.3.5 Get Extended Memory Array 7-211
7.3.6 Release FHD Port 7-212
7.3.7 Reserve FHD Port 7-212

Contents

Page

A MPX-32 Device Access A-1

B System Services Cross-Reference B-1

C MPX-32 Abort and Crash Codes C-1

D Numerical Information D-1

E Powers of Integers E-1

F ASCII Interchange Code Set F-1

G IOP/MFP Panel Mode Commands G-1

H Standard Date and Time Formats H-1

I Compressed Source Format I-1

J Map Block Address Assignments J-1

K Control Switches K-1

L Data Structures L-1

Glossary GL-1

Index IN-1

List of Figures

Figure		Page
1-1	MPX-32 Processors and Utilities	1-2
1-2	Hardware/Software Priorities	1-4
2-1	Nonbase Mode Nonshared Task Address Space	2-4
2-2	Nonbase Mode Shared Task Address Space	2-5
2-3	Base Mode Shared Task Address Space	2-6
2-4	Task Service Area (TSA) Structure	2-9
3-1	Sample Allocation of Common Memory Partitions and Common Code	3-22
3-2	Extended MPX-32 Physical Memory	3-25
3-3	Extended MPX-32 Program Flow Control	3-27
3-4	Tasks' Logical Address Space Using Extended MPX-32	3-31
3-5	Task's Logical Address Space Using the EXTDMPX Directive With TSA Keyword	3-36
4-1	Volume Format	4-21
4-2	A Sample Hierarchical Directory Structure	4-25
4-3	Locating a File on a Volume	4-27

List of Tables

Table		Page
1-1	CONCEPT/32 Trap Vectors	1-5
1-2	CONCEPT/32 Interrupt Vectors	1-6
1-3	MPX-32 Device Support	1-23
2-1	Nonbase Mode vs. Base Mode	2-2
2-2	MPX-32 State Queues	2-13
2-3	Task Interrupt Operation/Services Summary	2-35
2-4	H.IP0F Register Fixup	2-36
2-5	Task Termination Sequencing (EXIT, ABORT, and DELETE)	2-47
2-6	MPX-32 Faults/Traps and Miscellaneous Interrupts	2-52
3-1	Static versus Dynamic Shared Memory	3-18
3-2	Memory Partition Applications for Nonbase Mode Tasks	3-19
4-1	File Access Modes and Conditions	4-37
5-1	Assign/Open Resource Allocation Matrix	5-3
5-2	Multivolume Magnetic Tape Data Transfers Between Different Operating Systems	5-22
5-3	Disk Description Table	5-27
5-4	MPX-32 Device Type Codes and Mnemonics	5-31
5-5	Assign/Open Block Mode Determination Matrix	5-35
5-6	EOF and EOM Description	5-37
5-7	Type Control Parameter Block	5-42
5-8	Execute Channel Program FCB Format	5-46
5-9	Notification Packet Layout for PPCI Receiver	5-47
5-10	Permanent and Temporary File Resource Create Block (RCB)	5-59
5-11	Directory Resource Create Block (RCB)	5-62
5-12	Nonbase Mode Memory Partition Resource Create Block (RCB)	5-63

Documentation Conventions

Conventions used in directive syntax, messages, and examples throughout the MPX-32 documentation set are described below.

Messages and Examples

Text shown in `this distinctive font` indicates an actual representation of a system message or an example of actual input and output. For example,

```
VOLUME MOUNT SUCCESSFUL
```

or

```
TSM>!ACTIVATE MYTASK
TSM>
```

Lowercase Italic Letters

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

$NOTE *message*

means replace *message* with the desired message. For example,

```
$NOTE 10/12/89 REV 3
```

In system messages, lowercase italic 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 the input required to execute that directive. Uppercase bold letters indicate the minimum that must be entered. For example,

$ASSIGN *lfc* **TO** *resource*

means enter $AS or $ASSIGN followed by a logical file code, followed by TO and a resource specification. For example,

```
$AS OUT TO OUTFILE
```

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

```
TERMDEF HAS NOT BEEN INSTALLED
```

Brackets [ ]

An element inside brackets is optional. For example,

$CALL *pathname* [*arg*]

means supplying an argument (*arg*) is optional.

Multiple items listed within brackets means enter one of the options or none at all. The choices are separated by a vertical line. For example,

$SHOW [**CPU**TIME | **JOB**S | **USE**RS]

means specify one of the listed parameters, or none of them to invoke the default.

Items in brackets within encompassing brackets or braces can be specified only when the other item is specified. For example,

BACKSPACE **F**ILE [[**FIL**ES=] *eofs*]

indicates if *eofs* is supplied as a parameter, FIL= or FILES= can precede the value specified.

Commas within brackets are required only if the bracketed element is specified. For example,

LIST [*taskname*][,[*ownername*] [,*pseudonym*]]

indicates that the first comma is required only if *ownername* and/or *pseudonym* is specified. The second comma is required only if *pseudonym* is specified.

Braces { }

Elements listed inside braces specify a required choice. Choices are separated by a vertical line. Enter one of the arguments from the specified group. For example,

[**BLO**CKED={**Y** | **N**}]

means Y or N must be supplied when specifying the BLOCKED option.

Horizontal Ellipsis ...

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

$DEFM [*par*] [,*par*] ...

means one or more parameters (*par*) separated by commas can be entered.

Vertical Ellipsis

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

```
$DEFM SI,ASSEMBLE,NEW,OP
    .
    .
    .
$IFA %OP ASSM
```

means one or more directives have been omitted between the $DEFM and $IFA directives.

Parentheses ( )

In directive syntax, parentheses must be entered as shown. For example,

(value)

means enter the proper value enclosed in parentheses; for example, (234).

Special Key Designations

The following are used throughout the documentation to designate special keys:

<ctrl>	control key
<ret> or <CR>	carriage return/enter key
<tab>	tab key
<break>	break key
<bck>	backspace key
<del>	delete key

When the <ctrl> key designation is used with another key, press and hold the control key, then press the other key. For example,

<ctrl>C

means press and hold the control key, then press the C.

Change Bars

Change bars are vertical lines (|) appearing in the right-hand margin of the page for your convenience in identifying the changes made in MPX-32 Revision 3.5.

When an entire chapter has been changed or added, change bars appear at the chapter title only. When text within figures has changed, change bars appear only at the top and bottom of the figure box.

1 Introduction

1.1 System Description

The Mapped Programming Executive (MPX-32) is a disk-oriented, multiprogramming operating system that supports concurrent execution of multiple tasks in interactive, batch, and real-time environments. MPX-32 provides memory management, terminal support, multiple batch streams, and intertask communication.

MPX-32 uses the SelMAP to completely support the 16MB physical address space of the CONCEPT/32 computers. Each task executes in a unique address space that can be expanded under task control up to 2MB of memory on the 32/87, or 16MB on the 32/67, 32/97 and 32/2000. An integrated CPU scheduler and a swap scheduler provide efficient use of main memory by balancing the in-core task set based on time-distribution factors, software priorities, and task state queues. The SelMAP is used to perform dynamic relocation of tasks during inswap.

Tasks operating under MPX-32 can be activated and/or resumed by hardware interrupts, system service requests, interactive commands, job control directives, or by the expiration of timers. Multiple copies of a task can be executed concurrently in interactive, batch, or real-time environments. Through its various scheduling capabilities, MPX-32 provides the flexibility needed to adapt system operation to changing real-time conditions.

The MPX-32 software package is composed of various software modules including the resident operating system (I/O Control System (IOCS), CPU and swap schedulers, Resource Allocator, Volume Management module, reentrant system services, and device and interrupt handlers), a Terminal Services Manager (TSM), a system generator (SYSGEN), and utilities such as a Volume Formatter and Volume Manager. Figure 1-1 describes the system nucleus, processors, and utilities.

The Internal Processing Unit (IPU) is a second central processor designed to work with the CPU to increase system throughput. The IPU is attached to the SelBUS like the CPU and shares all memory (including the resident operating system area) with the CPU. The IPU's function is to execute user task level code in parallel with CPU operation. (The IPU is optional hardware and must be specified during SYSGEN for use on a system.)

To avoid contention between the IPU and CPU, there are IPU limitations. IPU cannot:

- communicate with peripherals (perform I/O)
- process all supervisor call (SVC) system services
- execute interrupt control instructions

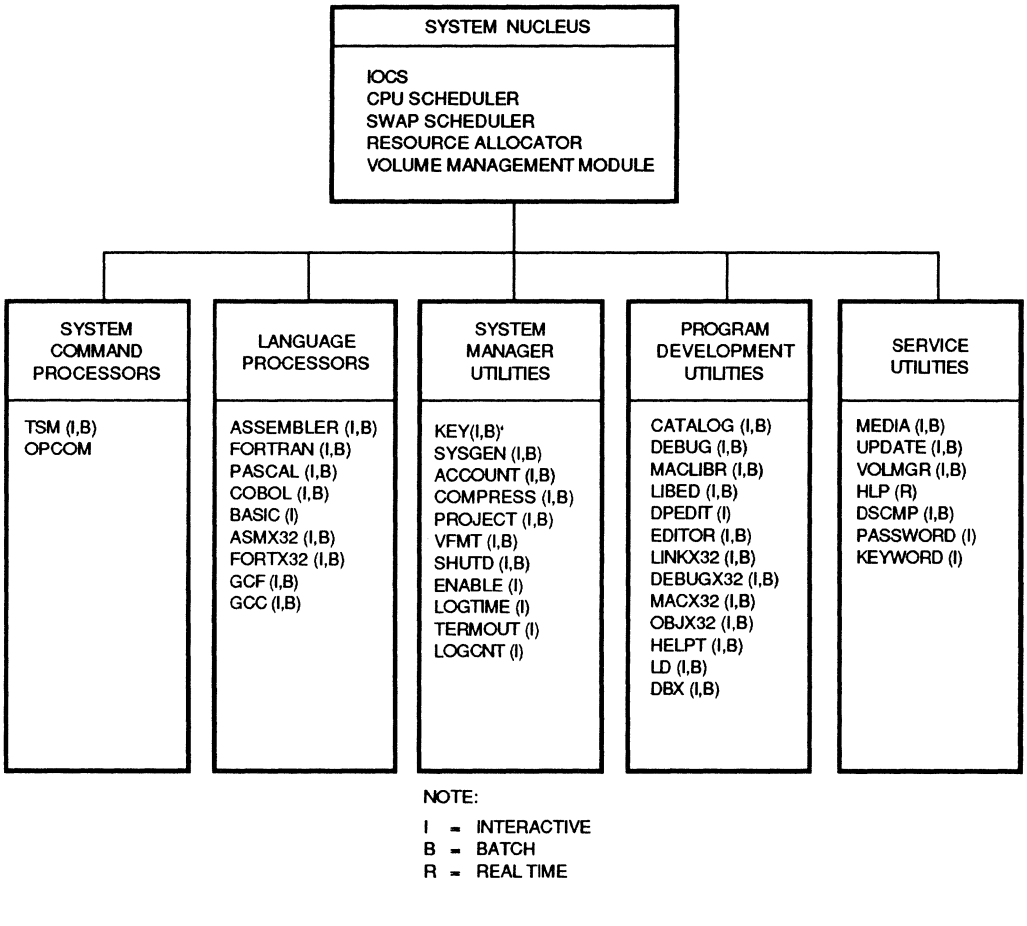


Figure 1-1
MPX-32 Processors and Utilities

Therefore, the IPU and CPU manage task execution transparently around the IPU limitations. For example, if the IPU is executing a task and encounters a service it cannot perform, a trap is sent to the CPU, the CPU takes over execution of the task at that point, and the task remains in the CPU until completion or reselection for IPU execution.

MPX-32 standard features include:

- full support for 16MB physical addressability of the CONCEPT/32 computers
- up to 255 tasks executing concurrently
- 64 software priority levels, 10 of which are time distributed
- servicing of all standard (extended I/O (XIO)) peripheral devices
- standard handlers for interrupts and traps
- intertask communications, including send/receive
- intertask shared memory partitions, such as Global Common and Datapool
- dynamic allocation and deallocation of memory and peripherals
- multiple batch streams, including multiple spooled input and output queues
- wait and no-wait I/O capabilities, including automatic blocking, buffering, and queueing
- terminal support for up to 64 devices, including device-independent operation and an extensive repertoire of online commands
- automatic task reentrancy through separation of pure code and data areas
- reentrant system services available to all tasks
- several levels of system security, including access restrictions based on task ownership
- file management, assignment, and security
- up to 245 logical files (files or devices) opened concurrently per task if both static and dynamic assignments are used
- project accounting capability
- transparent support of the IPU
- automatic mounting of public volumes at IPL

MPX-32 uses hardware and software priorities for scheduling and executing tasks. Figure 1-2 shows the various MPX-32 software elements and the hardware and software priority levels that are assigned to each.

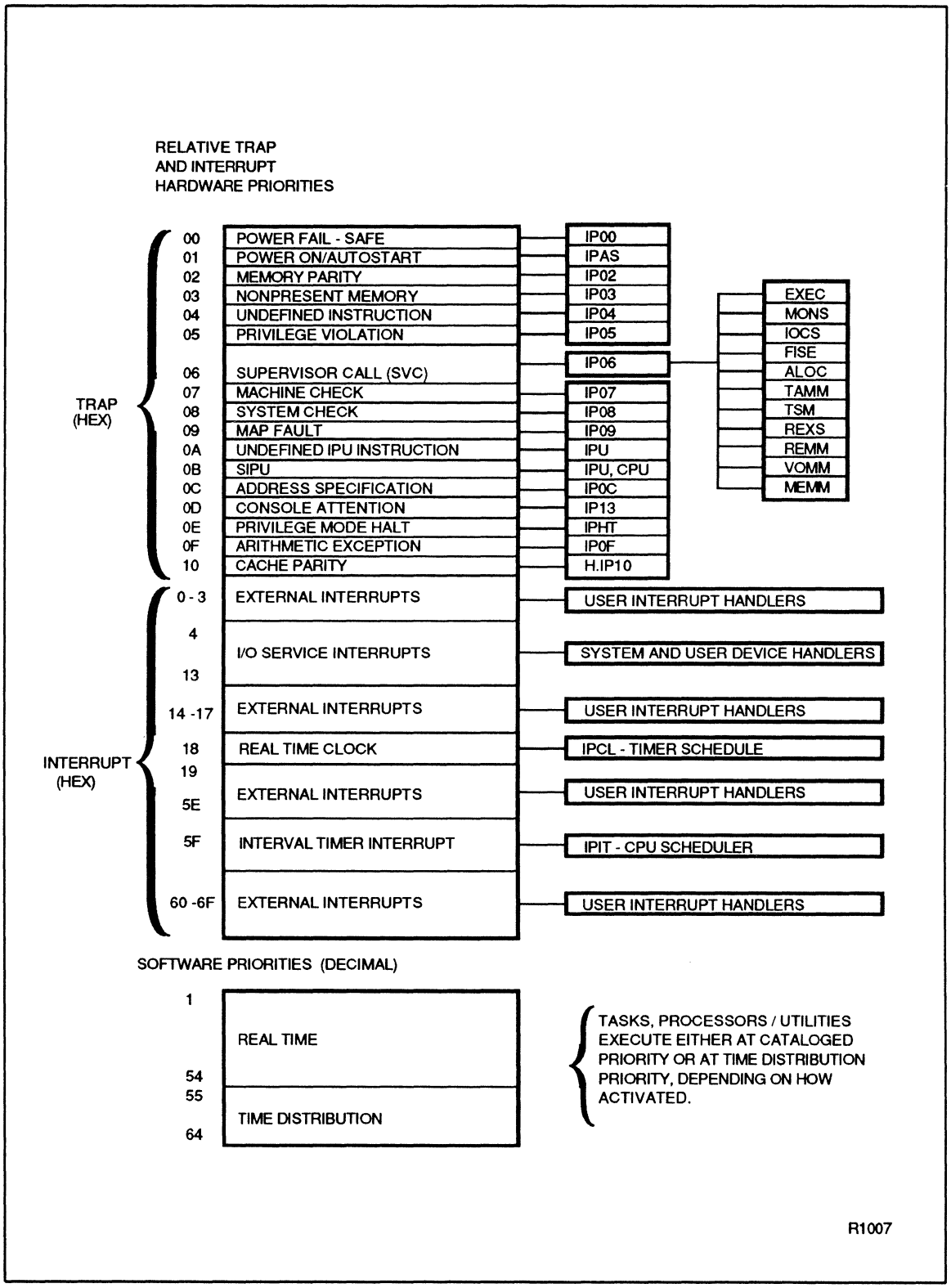


Figure 1-2
Hardware/Software Priorities

1.1.1 Hardware Interrupts/Traps

The CONCEPT/32 computers support up to 96 hardware interrupts and traps. See Tables 1-1 and 1-2 for a description of CONCEPT/32 trap and interrupt vectors. The exact number in a particular system depends on the user's requirements and the number of peripheral devices in the configuration.

The highest hardware priority levels in the system are reserved for the basic system integrity interrupts and traps. These include the power fail/power up traps and the system override interrupts and traps. Lower levels are used for the I/O transfer interrupts, memory parity trap, console interrupt, and I/O service interrupts.

The next lower group of interrupts and traps are used for exceptional conditions, supervisor call requests, and the real-time clock. The exceptional conditions include nonpresent memory trap, undefined instruction trap, privilege violation trap, and arithmetic exception interrupt.

All lower hardware priority levels are used for external interrupts. User tasks can be connected directly or indirectly to the external interrupts.

Table 1-1
CONCEPT/32 Trap Vectors

Relative priority	Default Trap Vector Location (TVL)		Trap Condition
	CPU	IPU	
00	80	20	Power fail trap (power down)
01	84	24	Autostart trap (power up)
02	88	28	Memory parity trap
03	8C	2C	Nonpresent memory trap
04	90	30	Undefined instruction trap
05	94	34	Privilege violation trap
06	98	38	Supervisor call trap
07	9C	3C	Machine check trap
08	A0	40	System check trap
09	A4	44	MAP fault trap
0A	A8	48	Not used
0B	AC	4C	Undefined IPU instruction trap
0C	B0	50	Address specification trap
0D	B4	54	Console attention trap
0E	B8	58	Privilege mode halt trap
0F	BC	5C	Arithmetic exception trap
10	C0	60	Cache memory parity trap (all supported CONCEPT/32 computers except the 32/2000)
11	C4	–	Demand page fault trap (CONCEPT 32/2000 only)

Table 1-2
CONCEPT/32 Interrupt Vectors

Relative Priority	Default Interrupt Vector Location (IVL)	Interrupt Condition
00	100	External/software interrupt 0
01	104	External/software interrupt 1
02	108	External/software interrupt 2
03	10C	External/software interrupt 3
04	110	I/O channel 0 interrupt
05	114	I/O channel 1 interrupt
06	118	I/O channel 2 interrupt
07	11C	I/O channel 3 interrupt
08	120	I/O channel 4 interrupt
09	124	I/O channel 5 interrupt
0A	128	I/O channel 6 interrupt
0B	12C	I/O channel 7 interrupt
0C	130	I/O channel 8 interrupt
0D	134	I/O channel 9 interrupt
0E	138	I/O channel A interrupt
0F	13C	I/O channel B interrupt
10	140	I/O channel C interrupt
11	144	I/O channel D interrupt
12	148	I/O channel E interrupt
13	14C	I/O channel F interrupt
14	150	External/software interrupts
.	.	.
.	.	.
.	.	.
17	15C	External/software interrupts
18	160	Real-time clock interrupt
19	164	External/software interrupts
.	.	.
.	.	.
.	.	.
5E	278	External/software interrupts
5F	27C	Interval timer interrupt
60	280	External/software interrupts
.	.	.
.	.	.
.	.	.
6F	2BC	

1.1.2 Software Interrupt System

MPX-32 provides 64 software priority levels for controlling the user's application. All system scheduling is performed by priority. Multiple tasks can be assigned to any priority level, thereby achieving a high level of multiprogramming versatility. The software priority levels are used by the Resource Allocator for peripheral and memory allocation, by the I/O supervisor for the queueing of I/O requests, and by MPX-32 whenever CPU control is allocated.

1.1.3 Task Priority Levels

Priorities 55 to 64 are time-sliced to provide for round-robin time distribution among tasks of the same priority. Priorities 1 to 54 are not time distributed. A task's cataloged priority is altered based on its eligibility to run. For example, a task's priority is boosted when an I/O operation is completed and restored after a minimal time quantum. Priority migration ensures maximum response to real-time events.

1.1.4 Supervision and Allocation

CPU scheduling is maintained through a set of state queues including the priority state chains and such execution states as suspended, queued for memory, queued for peripheral, I/O wait, etc. Each CPU dispatch queue entry defines all scheduling attributes of a single task. The entry typically migrates among the state queues as the task's execution eligibility changes. These state chains are also used by the swap scheduler to select candidates for swapping.

The CPU scheduler is invoked whenever a scheduling event occurs. Scheduling events include:

- external interrupts
- I/O completion
- timer expiration
- resource deallocation
- system service completion

IPU scheduling is maintained through state queues consisting of biased tasks (C.RIPU) scheduled in addition to MPX-32 normal state queues for nonbiased tasks (SQRT through SQ64). Any biased tasks are prioritized among themselves, and are scheduled for execution based on priority. Any nonbiased tasks are also prioritized among themselves, and are scheduled for execution according to priority. If a nonbiased task waiting for execution has a higher priority level than a biased task also waiting for execution, the nonbiased task is selected for execution.

An optional scheduling algorithm is available to boost the priority of IPU tasks and allow them to run in the CPU.

1.1.5 Memory Allocation

The unit of memory allocation is a map block, which is 2KW on the CONCEPT/32 computer. Memory is allocated to tasks as needed. All tasks are loaded discontiguously into a whole number of physical map blocks, utilizing the SelMAP to create their contiguous logical address space. No partial map blocks are allocated.

The MPX-32 memory allocation scheme allows tasks to dynamically expand and contract their address space by system service calls.

The unit of memory protection is called a protection granule and is 512W. Thus, it is possible to have protected data areas within a map block. On a CONCEPT 32/2000 with mapped out image, the unit of memory protection is 2KW.

1.1.5.1 Dynamic Allocation

Dynamic allocation and deallocation are performed by the allocate and deallocate system services. These services can be used to dynamically allocate and deallocate any peripheral device, permanent and temporary disk files, or the system listed output (SLO) and system binary output (SBO) files. By allocating peripheral devices dynamically, each task has exclusive use of a peripheral only during the time required to perform the task's I/O. Therefore, when peripherals are not allocated, other tasks can use them on an as-needed basis.

Because the allocation of system-wide peripheral devices that are requested dynamically cannot be guaranteed, a task must be prepared to accept a denial return.

A task requesting additional memory is automatically queued until the memory can be allocated. For peripherals and file space, the caller can optionally queue for allocation or take alternative action.

1.1.6 File Management

In the MPX-32 operating environment, files are used in several ways. Permanent files are created for user programs, user data, and system programs. Temporary files provide system scratch storage, user scratch storage, and system output data storage for the system printer. Separation is maintained among files belonging to different users.

The file management system for MPX-32 consists of the resident Volume Management Module and the nonresident Volume Manager. Together, they supervise all file space on the disks.

1.1.6.1 Permanent Files

Residing in disk storage, permanent files are defined by entries in a directory which specify each file's name, binary creation date and time, absolute block number of resource descriptor, resource ID flag and type and other directory entry control information. Permanent files remain defined to the operating system until they are explicitly deleted.

Permanent files can be accessed by multiple tasks for both input and output. Access permanent files by pathname. To locate the directory entry for a permanent file, MPX-32 translates the file name characters to a specific location in the directory. For a complete description of pathnames, see Chapter 4.

Permanent files are classified as either fast or slow depending on the speed at which their directory entries can be located. A fast permanent file is one whose entry can be located with one disk access. Slow permanent files require two or more disk accesses to find each file's entry.

1.1.6.2 Temporary Files

Temporary files are files whose definitions are eliminated from the system upon completion of the task requiring the space. Temporary file space is allocated and deallocated by the Volume Management Module which is responsible for maintaining space allocation maps for all available disks. Temporary files are typically used for either system or user scratch storage.

1.1.6.3 Random Access Files

Any disk file may be accessed randomly by record number through standard IOCS calls. The user sets a bit and specifies the relative disk block number in a file control block (FCB) to utilize this feature.

1.1.6.4 Disk File Protection

File protection mechanisms are available to prevent unauthorized access to and deletion of permanent files. Protection of individual files can be specified when the files are created. User files can also be protected on a per user basis. If a key is associated with an owner name in the M.KEY file, it must be entered at logon. Specific access rights are defined for each file by owner, project group, and other.

1.1.6.5 Dedicated System Files

To increase system throughput and minimize I/O delay time, IOCS supports disk buffered I/O by using special system files. Four dedicated file codes exist in the system. One file code is for buffered system input (SYC), two file codes are for buffered system output (SLO, SBO), and one file code is for a system object file (SGO). A system file can be assigned to a file code in the same manner a device is assigned to a file code.

1.1.6.6 Multiprocessor Files

MPX-32 allows tasks executing in separate system environments to concurrently access selected files. The operating system maintains resource integrity on these files within the scope of volume management described in Chapter 4 of this manual. These files must reside on a volume accessible by a multiported device.

1.1.7 System Services

MPX-32 offers resident system service routines that can perform frequently required operations with maximum efficiency. Using the Supervisor Call instruction, tasks running in batch, interactive, or real-time environments can call these routines.

All system service routines are re-entrant. Thus, each service routine is always available to the currently active task.

The system service routines are standard modular components of MPX-32. However, the open-ended design of the system gives each user freedom to add any service routines required to tailor MPX-32 to a specific application.

1.1.8 Input/Output Operations

The Input/Output Control System (IOCS) provides I/O services that relieve the programmer of detailed chores. While keeping software overhead to an absolute minimum, IOCS receives and processes all I/O requests for both user and tasks. It performs all logical error checking and parameter validation. IOCS also logically processes all I/O operations and assigns I/O control to the appropriate device handler. In turn, the device handler executes the I/O data exchange, processes service interrupts, and performs device testing.

I/O operations for MPX-32 include the following general capabilities: direct I/O, queued I/O requests, device independent I/O, device interchangeability, device reassignment, and disk-buffered (blocked) I/O.

1.1.8.1 Direct I/O

I/O can be issued directly to acquire data at rates which prohibit the overhead of IOCS. Mechanisms are provided in IOCS to ensure that no conflict occurs with IOCS file operations. The interface facilities provided in IOCS for direct I/O enable a task to gain exclusive use of an I/O channel.

1.1.8.2 Device-Independent I/O

Normal I/O operations in the system occur to and from user-specified logical file codes. These file codes are assigned and reassigned to the physical device where the I/O commands are ultimately routed.

1.1.8.3 Logical File Codes

The user logical file code consists of one to three ASCII characters. For each file code defined and referenced by a user task, there is an entry in a file assignment table (FAT). The FAT entry describes the device controller channel and the device the file is assigned to. For a disk that is a shared device, additional addressing information is provided for complete identification of the file. Each user task is allowed a maximum of 245 static and dynamic logical file assignments.

1.1.8.4 File Access

IOCS supports both random and sequential file access. Random or sequential access is specified by the user. All files assigned to devices other than disk are considered sequential. A file assigned to disk may be referenced by both random and sequential transfers. Attempts to perform a write operation on a file specified as read-only, or attempts to circumvent disk file protection and security, are aborted.

1.1.9 Communications Facilities

MPX-32 offers complete facilities for communications between individual users, internal system elements, user tasks, and the operator and the system. Users communicate with one another by sharing permanent files, shared images, the communication region, Global Common and Datapool partitions, and job status flags which can be set and interrogated by system service routines. Tasks communicate with one another by messages or run requests.

1.1.9.1 Intertask Messages

Tasks can establish message receivers for intertask communication. Messages are buffered by MPX-32 in memory pool until the receiving task is eligible to receive. The receiving task is interrupted asynchronously and optionally responds to the sender. The sender optionally waits for a reply or elects to be interrupted asynchronously by a response. Messages can be queued to an arbitrary depth.

1.1.9.2 Run Requests

A task can send a run request to any other task. A run request is similar to a message, except that with a run request, the receiver may not yet be in execution. In such cases, the receiving task is activated before the message is queued. The receiving task can process run requests at any time.

1.1.9.3 Global Common

Global Common is an area of memory that many programs can access by using symbolic names to identify specific storage cells. In this respect, Global Common is comparable to local common. Unlike local common, however, access to Global Common is not restricted to programs within a single task. Rather, programs belonging to many independent tasks can freely access the same data and exchange control information within the Global Common area.

1.1.9.4 Shared Images

A shared image allows base mode tasks to share both code and data, for example, shared subroutines and common data partitions. A shared image is built on disk by the base mode linker (LINKX32) and is loaded into memory upon inclusion by a task. Nonbase mode tasks can include the shared image as an initialized dynamic data partition. Shared images are distinct from static and dynamic common in that the memory is initialized with data from disk.

1.1.9.5 Datapool

Like Global Common, Datapool is an area of memory that many tasks can access using symbolic references. In addition to providing all the advantages of Global Common, Datapool provides a much wider range of structuring flexibility. For example, where global common symbolic references must follow the same order as the locations of the data in memory, symbolic references to Datapool may be entirely independent of the actual positioning of data within the memory area.

1.1.9.6 Internal Communications

Internal system elements communicate through temporary files, system queues, and the system communications region. The system communications region occupies approximately 2KW of lower memory. It contains information common to all system modules and processors.

1.1.10 Trap Processors

Trap processors are entered when any exceptional condition trap occurs. Certain traps indicate task errors, such as a reference to nonpresent memory, a privilege violation, or execution of an unimplemented instruction. These traps cause the violating task to be aborted. When the arithmetic exception trap occurs, the overflow condition is noted for use by the task in execution.

1.1.11 Timer Scheduler

The timer scheduler schedules events such as task activation, task resumption, flag setting and resetting, and interrupt activation on a timed basis.

1.1.12 Time Management

Time is kept in two different formats. The system maintains the time as a count of clock interrupts, and the date as an ASCII constant. To allow for easy time stamping of resources with the file system capabilities, time is also kept as a binary count of 100 microsecond units since midnight, and the date as the binary number of days since January 1, 1960. See Appendix H for more details.

When entering the date and time, the user can specify daylight savings time and a correction factor for time zone. These features are provided for the user who wants the system to use a standard time base, such as GMT, for system operations, and to display values of date and time in local time. For example, if a user states that local time is `10:00:00`, daylight savings time is in effect, and there is a two-hour correction for time zone, the time kept by MPX-32 indicates `07:00:00`. The correction factor is kept so any user access of time indicates the local value.

Another feature allows the use of the international date format for entering the date. Instead of entering `10/17/80`, `17OCT80` can be entered. The date is always displayed in the same format as it is entered at IPL time.

1.1.13 System Nonresident Media Mounting Task (J.MOUNT)

J.MOUNT mounts both formatted volumes and unformatted media. J.MOUNT is normally in the Waiting for Run Request (RUNW) queue. When a task requires a volume to be physically mounted or dismounted, a run request is sent to J.MOUNT. J.MOUNT then interacts with the operator through the system console to complete the mount or dismount process.

1.2 System Command Processors

The Terminal Services Manager (TSM) and the interactive Operator Communications (OPCOM) command processor provide access to MPX-32 interactive, batch, and real-time processing environments.

1.2.1 Terminal Services Manager (TSM)

TSM provides interactive, time-shared access to the MPX-32 system for terminals connected either through ALIM or ACM controllers. It allows the user to:

- logon to MPX-32
- access any MPX-32 processor
- communicate with online users or the operator
- account for use of computer resources
- specify and pass parameters to interactive and batch tasks
- automate a series of tasks into a job, or submit a stream of jobs
- nest directive files
- construct loops to control processing in directive files
- request assignment of any MPX-32 resource
- specify alternative actions conditionally
- logoff from MPX-32

1.2.2 Operator Communications (OPCOM)

OPCOM provides commands that set up the system for optimum response to changing conditions. Using OPCOM directives, users can:

- list the status of all queues, tasks, I/O controllers, and mounted volumes
- control spooled print and punch output
- hold and continue execution of tasks
- activate and abort tasks
- connect tasks to interrupts
- establish resident and nonresident tasks
- display time-of-day clock
- create and delete timer scheduler queue entries
- delete allocation queue entries
- enable, disable, and initiate hardware interrupts
- reserve devices, release them, and place them off-line or online
- change the assignment of the system input device, the SGO file, and the destination of the SLO and SBO spooled output files
- initiate the reading of the batch stream
- issue system debugging commands
- dump physical memory to the console

1.2.3 Batch Processing

Batch processing consists of spooling batch jobs to disk, interpreting job control statements, and directing listed and binary spooled output to destination files and devices. Multiple jobs are processed concurrently within limits established by SYSGEN and the availability of computer resources. Tasks that use batch processing compete with each other and with nonbatch tasks for computer resources under standard MPX-32 allocation algorithms.

Each job is spooled to a separate system control (SYC) disk file prior to processing. Jobs can be spooled to SYC files from card, magnetic tape, and paper tape peripheral devices, and from blocked, temporary, and permanent disk files. The OPCOM BATCH directive can be used to initiate spooling from peripheral devices and permanent files. The batch job from entry system service (M.BATCH) is used by TSM, and the Text Editor and can be invoked by a task to initiate spooling from permanent and temporary disk files. The TSM $BATCH or $SUBMIT directives can also be used to submit batch jobs.

Job sequence numbers show the order that jobs are entered and uniquely identify each job and its tasks.

When a job completes, its spooled listed and binary output is automatically routed to usable peripheral devices if no particular device(s) or permanent file(s) are specified for the job. Usable devices for automatic selection are specified by SYSGEN and OPCOM directives. Spooled output destination devices include line printers, card punches, magnetic tapes, paper tapes, and disk files. Spooled output is selected for processing based on the software priority of jobs and, within a given priority, on the order in which jobs complete processing.

1.3 Program Development Utilities

MPX-32 supports both nonbase and base mode programs. Refer to the Task Structures section of Chapter 2 for the nonbase and base task differences. Nonbase and base modes cannot be mixed; therefore, separate program development utilities exist for each mode.

Of the following utilities, only VOLMGR and J.VFMT are included on the MPX-32 Master SDT.

1.3.1 Task Cataloging (CATALOG)

Use the cataloger to catalog permanent nonbase mode load modules that execute as tasks on the MPX-32 system. During cataloging, relocatable object modules produced by the assembler or compilers are loaded and linked internally and externally to library subroutines. The linked body of code thus produced is then sent to a selected permanent file in relocatable or absolute format. In addition, the cataloger places a preamble on this file. This preamble contains a summary of the resources required by the task, such as memory, permanent files, and peripheral devices, and defines special task characteristics (shared, resident, etc.). Once created, a task is known to the system by the name of the permanent file where it resides. The task can then be activated, saved, restored, or otherwise operated on by specifying its name in the appropriate job control statement, system service call, or terminal directive.

1.3.1.1 Privilege

Whether a task is privileged or unprivileged can be defined by cataloger directives. The ability to specify a privileged operation for a task can be restricted by owner name.

By specifying whether tasks are privileged or unprivileged, users can control system security. Tasks designated to run privileged are free to execute any instruction in the instruction repertoire. They also have read/write access to all memory locations.

1.3.1.2 Overlays

For efficient use of memory, the cataloger provides the user with facilities for dividing large nonbase mode programs into overlays. The main program segment, the root, and the overlay segments can be cataloged in relocatable format. Individual overlays can be cataloged separately, permitting the user to modify or replace any overlay without disturbing any of the others. Flexible symbol linkage is provided between the root and its associated overlays and between individual overlays of various levels.

1.3.2 Task Debugger (AIDDB)

The task debugger is a directive-oriented processor that debugs a single, cataloged, nonbase mode user task. It can be accessed with a DEBUG directive in TSM, with a $DEBUG statement in batch, by coding an M.DEBUG service call within the cataloged task, or by using the break key after a task has been activated with TSM, in which case TSM provides the option of calling M.DEBUG.

If the task the debugger is connected to has a shared CSECT, the debugger must be attached at task activation (by the DEBUG directive in TSM or $DEBUG statement in batch). The shared CSECT task is then loaded as multicopied and breakpoints set in the CSECT do not impact other users of the shared CSECT.

Using AIDDB directives, users can:

- trace task execution
- set debugging traps within the task
- display and/or alter contents of the task's logical address space, general purpose registers, etc.
- watch for privileged task entry into the operating system or other areas of memory not usually accessed even by a privileged task
- perform other operations that facilitate task debugging

1.3.3 Macro Assembler (ASSEMBLE)

The Macro Assembler translates nonbase assembler directives and source code into binary instructions for the CONCEPT/32 CPU.

1.3.4 Macro Library Editor (MACLIBR)

With the Macro Library Editor, nonbase mode macros that are used frequently can be placed in a macro library where they are available for use by the Macro Assembler. During execution, the Macro Library Editor transfers the macros from the source input file to the macro library file. The macros entered into the library are listed on an output file.

1.3.5 Subroutine Library Editor (LIBED)

The Subroutine Library Editor provides facilities for creating and modifying the nonbase mode system subroutine library and any number of user subroutine libraries. The user is provided with a listing of directives, module names, external definitions, the quantity of library and directory space remaining on the disk, and the modules that were specified for deletion but were not located in the library.

1.3.6 Datapool Editor (DPEDIT)

The Datapool Editor provides the ability to create and maintain dictionaries for access to static or dynamic Datapool common memory partitions.

1.3.7 Text Editor (EDIT)

The Text Editor provides directives for building and editing text files, merging files or parts of files into one file space, copying existing text from one location to another, and, in general, for performing editing functions familiar to users of interactive systems.

EDIT is typically used to create source files and to build job control files and general text files. A job file built in the editor can be copied directly into the batch stream using the editor BATCH directive.

1.3.8 Volume Manager (VOLMGR)

The Volume Manager creates or deletes permanent disk file space, special global partitions, and/or a datapool partition (one that can be dynamically allocated in memory when required by tasks). A primary use is to provide system and user permanent file backup.

1.3.9 Volume Formatter (J.VFMT)

The Volume Formatter formats volumes (disks). It can operate on a fully functional MPX-32 system or a starter system by the SDT.

1.3.10 Assembler/X32 (ASMX32)

The Assembler/X32 translates base mode assembler directives and source code into binary base mode instructions for the CONCEPT/32 CPU.

1.3.11 Macro Librarian/X32 (MACX32)

The Macro Librarian/X32 builds and maintains base mode macro libraries that are accessed by the macro assembler/X32. Frequently used base mode macros can be placed in the macro libraries for easy access.

1.3.12 Object Librarian/X32 (OBJX32)

The Object Librarian/X32 provides facilities for creating and modifying user object libraries. The object libraries contain object files to be used in base mode programs. The object librarian provides a log of the number of object files entered, the names of the object files, when each file was entered, and the amount of available library space.

1.3.13 Linker/X32 (LINKX32)

The Linker/X32 creates permanent base mode load modules that can execute as tasks on MPX-32. During linking, object modules produced by the macro assembler/X32 or compilers are loaded and linked internally and externally to library subroutines. The linked body of code becomes an executable image.

1.3.14 Symbolic Debugger/X32 (DEBUGX32)

The Symbolic Debugger/X32 is a directive-oriented processor used to debug base mode executable images created by the Linker/X32. Using the debugger, users can:

- debug interactively, with debugger directives controlling the execution of the program
- access program locations (memory addresses) by using hexadecimal addresses, bases, or the global symbols defined in the source program — addresses are displayed either in hexadecimal format or relative to a base or global symbol
- display data in ASCII, hexadecimal, or instruction format
- execute program instructions one at a time, showing the result after each instruction is executed, or set traps to allow execution to proceed through many instructions to a designated program checkpoint
- define bases
- debug privileged programs
- print a record of the debugging session

1.4 Service Utilities

1.4.1 Source Update (UPDATE)

The Source Update utility provides facilities for revising source files. It permits the user to enter new files, as well as to update existing files by adding, replacing, and deleting source statements. Input can be in standard or compressed format. Either format can be selected for the output file. Source Update can also produce a listing of the control stream as it generates the output file.

1.4.2 Media Conversion (MEDIA)

The Media Conversion utility performs functions ranging from card duplication to merging multiple media inputs into single or multiple media outputs. It provides media editing, media-to-media conversion, code conversion, media copying, and media verification. Rather than restricting the user to a fixed set of functions, the Media Conversion utility is controlled by a language of directives.

1.5 System Manager Utilities

1.5.1 M.KEY Editor (KEY)

KEY is a utility used to build an M.KEY file for the MPX-32 system. The M.KEY file specifies valid owner names on the system and optionally sets, for each owner name:

- a key and/or password to restrict access to the owner name during logon and to the user name when accessing files
- OPCOM indicators restricting the owner's use of OPCOM directives
- an indicator that prevents the owner from cataloging privileged tasks (tasks that use privileged system services or privileged variations of unprivileged system services)
- an indicator that prevents the owner from activating tasks cataloged as privileged
- default tab settings
- default working volume and directory specification
- default alphanumeric project names/numbers for accounting purposes

After KEY runs, only those owners established in the M.KEY file can logon to the system and access files.

1.5.2 MPX-32 System Start-up, Generation, and Installation (SYSGEN)

Users can install a starter system by booting from the master System Distribution Tape (SDT). Using the starter system, which is fully operational, a user-configuration of the system can be generated with the SYSGEN utility (running either interactively or in batch). An online RESTART directive is available to test user-configured systems. When a system has been tested, users can create their own SDT using the VOLMGR SDT directive.

When SYSGEN runs, 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 a dynamically acquired permanent disk file. Concurrent with this process, a listing of directives is built and a load map of the system is generated. The load map can be saved on a system symbol table file specified by the user with the SYMTAB directive and used subsequently in patching the system.

A system debugger can also be configured in the resident system image to assist in patching or debugging resident system code, including user interrupt and I/O handlers.

1.6 Libraries

1.6.1 Subroutine Libraries

Subroutine libraries can simplify the development of applications. Subroutines can be added, modified, or deleted. This permits one routine to be changed without having to reassemble or recompile all of the subroutines needed for a task. Only the task must be recataloged.

Subroutines on a subroutine library can be used by programs written in various languages, including Assembly. They are accessed as object modules when a task is cataloged. The subroutine library and directory for MPX-32 are called MPXLIB and MPXDIR. User subroutine libraries can be built and modified by the LIBED utility.

1.6.2 System Macro Libraries

Two macro libraries are supplied as part of the MPX-32 system. They are used only with programs written in assembly language. The first, M.MPXMAC, should be accessed when code that uses MPX-32 system services is assembled. The second, M.MACLIB, is used when code contains RTM monitor service calls. These macro libraries provide macros containing equates for MPX-32 communication region variables.

The user can expand, contract, or modify a macro library by using the MACLIBR utility.

1.6.3 Other

The Scientific Subroutine Library is optionally available. It contains math and statistical routines for scientific and engineering applications. A user group library is also available. It is provided by and for users.

1.7 Minimum Hardware Configuration

Minimum hardware requirements for MPX-32 operation on a CONCEPT/32 computer are as follows:

- 128KW memory
- magnetic tape (class F) or IOP floppy disk
- I/O processor (IOP) or Multi-Function processor (MFP) console
- extended I/O disk

The minimum configuration must also include the prerequisites required to support the items listed, for example, controllers, formatters, etc.

Devices supported by MPX-32 are listed in Table 1-3. Where appropriate, the code used to access a device is shown in parentheses. The code indicating the appropriate device, such as TY for a terminal on an ALIM, is used when accessing devices connected with a communications link.

Table 1-3
MPX-32 Device Support

Model Number	Description
1603	Vector Processor 3300*
1604	Vector Processor 6410*
2345	Real-Time Option Module
3050	Multiprocessor Shared Memory System (MS)2
7302	Reflective Memory Port (RMS) with WSC, RSC
7410	Analog Digital Interface (ADI)
8001	I/O processor
8002	Multi-Function Processor (MFP)
8031	Line printer/Floppy disc controller (LP)
8050	High Speed Tape Processor (HSTP) (XIO)
8055	Disc Processor II*
8060	Universal Disc Processor (UDP)
8064	High-Speed Disc Processor (HSDP)
8121	80MB sealed media disc processor subsystem
8130	80MB disc processor subsystem
8140	300MB disc processor subsystem
8150	675MB fixed module disc processor subsystem
8160	Cache disc accelerator
8174	Floppy disc with controller (FL)
8175	Dual floppy disc with controller (FL)
8210	High speed tape processor subsystem 75 ips 9-Track 1600/6250 bpi (M9)*
8211	High speed tape processor subsystem 125 ips 9-Track 1600/6250 bpi (M9)*
8212	High speed tape processor subsystem 125 ips 9-Track 800/1600/6250 bpi (M9)
8255	Buffered tape processor
8310	Band printer (300 lpm) (64 character) (LP)
8311	Band printer (600 lpm) (64 character) (LP)
8312	Band printer with form length select switch
8313	Band printer with VFU (300 lpm) (64 character) (LP)
8314	Band printer with VFU (600 lpm) (64 character) (LP)
8315	Band printer with VFU (1000 lpm) (64 character) (LP)
8356	Matrix Printer (80 col)
8357	Matrix Printer (130 col)
8371	Letter Quality Printer
8317	96-character option set
8410	Quarter-inch tape drive
8510	Eight-line asynchronous communication controller*

* This product is no longer available but remains supported by MPX-32 in existing installations.

Continued on next page

Table 1-3
MPX-32 Device Support (*Continued*)

Model Number	Description
8511	Asynchronous Communication Multiplexer (ACM)*
8512	Asynchronous Communication Multiplexer (ACM)*
8520	Synchronous Communications Multiplexer (SCM)
8610	Alphanumeric CRT (CT or TY)
8846	160MB disc processor subsystem
8856	340MB disc processor subsystem
9020	Low Speed Tape Processor (LSTP) (XIO)
9103	Extended (Class D) General Purpose Multiplexer Controller (GPMC)
9109	Synchronous Line Interface Module (SLIM)
9110	Asynchronous Line Interface Module (ALIM)
9116	Binary Synchronous Line Interface Module (BLIM)
9131	High Speed Data Interface II (HSD II)
9202	Teletypewriter (30 cps) (CT or TY)
9203	Alphanumeric CRT (95 character) (CT or TY)*
9223	Matrix printer (340 cps) (LP)
9225	Line printer (300 lpm) (64 character) (LP)*
9226	Line printer (600 lpm) (64 character) (LP)*
9237	Line printer (900 lpm) (64 character) (LP)*
9245	Line printer (260 lpm) (96 character) (LP)*
9246	Line printer (436 lpm) (96 character) (LP)*
9247	Line printer (600 lpm) (96 character) (LP)*
9460	Paper tape reader with controller (300 cps) (PT)
9462	Paper tape reader/spooler (300 cps) (PT)
9567	Low speed tape processor subsystem 45 ips 9-track 800 bpi (M9)*
9568	Low speed tape processor subsystem 45 ips 9-track 800/1600 bpi (M9)
9571	Low speed tape processor subsystem 75 ips 9-track 800/1600 bpi (M9)
9577	75 ips Master magnetic tape unit 9-track (M9)*

* This product is no longer available but remains supported by MPX-32 in existing installations.

2 Task Structure and Operation Overview

2.1 Task Identification

The user can identify tasks by task name or task number. The task name is the name of the load module or executable image file containing the task. The task number is a sequential 24-bit number concatenated with an 8-bit DQE index and is assigned when the task is activated. Task numbers are unique for each task in the system. If the task is multicopied, use the task number.

Each task is also associated with an owner. Valid owner names are specified in the M.KEY file, if it exists; otherwise, all owner names are valid. An owner can have any number of tasks with the same or different task names active on the system at any time.

In addition to the task numbers, each batch job is assigned a unique sequence number when the job is entered in the batch stream.

Active tasks can be listed by:

- task number
- owner name
- task name
- batch sequence number (if batch)
- pseudonym used by MPX-32 to further identify the task, e.g., by the terminal it is running on
- any combination of the above

The system provides the OPCOM LIST directives and the system service M.ID for listing any active task by specifying a unique combination of these attributes.

2.2 Task Structure

A task is structured to meet a user's particular requirements by defining the contents of a unique address space. A unique address space is a mapped logical address space whose maximum size varies, according to computer type. The unique address maximum executable code region size depends on whether the nonbase or base instruction set is being used. See Table 2-1.

Table 2-1
Nonbase Mode vs. Base Mode

	Nonbase Mode	Base Mode
Supported on	All CONCEPT/32 computers	All CONCEPT/32 computers
Maximum task size	2 MB (32/87) 16 MB (all others)	2 MB (32/87) 16 MB (all others)
Code/data size	0.5 MB	2 MB (32/87) 16 MB (All others)
Data-only size	1.5 MB (32/87) 15.5 MB (All others)	N/A
Name of shareable area	CSECT	Read-only
Name of nonshareable area	DSECT	Read/write
Created by	CATALOG	LINKER/X32
Exists on disk as	Load module	Executable image

All tasks activated on a 32/87 have a 2MB logical address space.

Base mode tasks activated on all other CONCEPT/32 computers have a logical address space of 2MB or 32KW plus the task size, whichever is larger. The automatic logical address space sizing can be overridden by the SET LAS LINKX32 directive or the TSM $SPACE directive.

Nonbase mode tasks activated on computers other than the 32/67 and 32/97 have a logical address space of 2MB unless overridden by the $SPACE TSM directive.

A unique address space contains a copy of MPX-32 and a task that can:

- be nonshared
- share re-entrant code and data with another task
- share memory (common storage or user defined use) with another task

The memory size minus the operating system size equals the maximum task size. The operating system size includes any static memory partitions and 4KW for use by the Volume Management Module.

Shared memory considerations are described in Chapter 3.

2.2.1 Nonbase Nonshared Tasks

This type of address space contains a single task including its task service area (TSA), its code section (CSECT — write protected memory containing code and pure data), and its data section (DSECT — read/write memory containing impure data). See Figure 2-1. Tasks which are not sectioned have only a DSECT, which contains the code and all data.

2.2.2 Base Nonshared Tasks

This type of address space contains a single task including its TSA, program stack, read/write image section, and read-only image section.

2.2.3 Multicopied Tasks

An owner or several owners can have tasks with the same name and the same load module active concurrently. This is accomplished by cataloging the task as multicopied. To communicate with multicopy tasks, the task number must be used.

2.2.4 Shared Tasks

When a task is created, the user can specify that a program section is to be shared. A program section, CSECT or read only, consists of code and pure data. This section is write protected and mapped into the logical address space of each copy of the task. A separate data section, DSECT or read/write, is mapped into each logical address space, as illustrated in Figures 2-2 and 2-3. Shared tasks are implicitly multicopied tasks.

2.2.5 Unique Tasks

Although only one copy of a task that is unique can be active on the system at a given time, the MPX-32 run request mechanism can be used to queue run requests to the task, so that as soon as one user stops executing, another can begin. For more information, see the Intertask Communication section of this chapter.

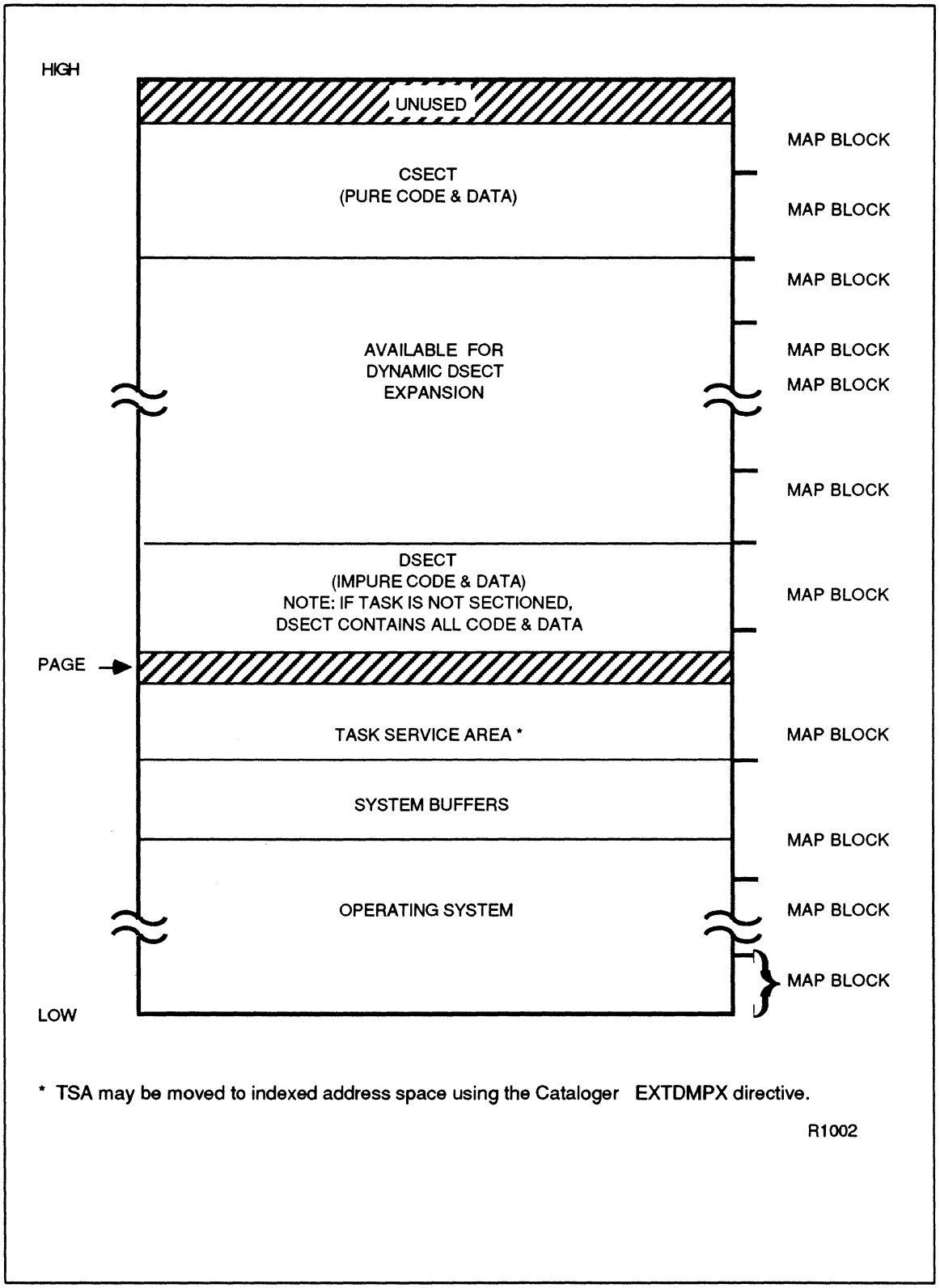


Figure 2-1
Nonbase Mode Nonshared Task Address Space

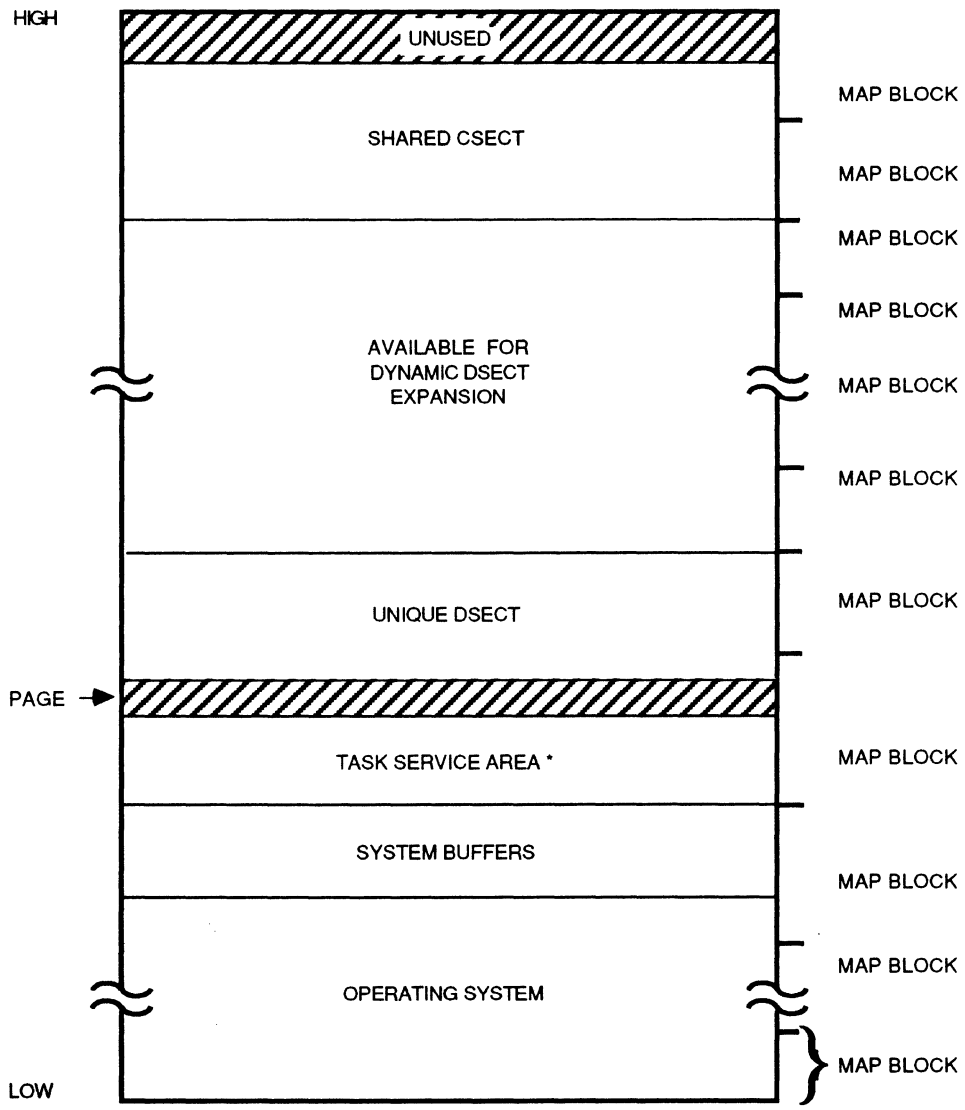


Figure 2-2
Nonbase Mode Shared Task Address Space

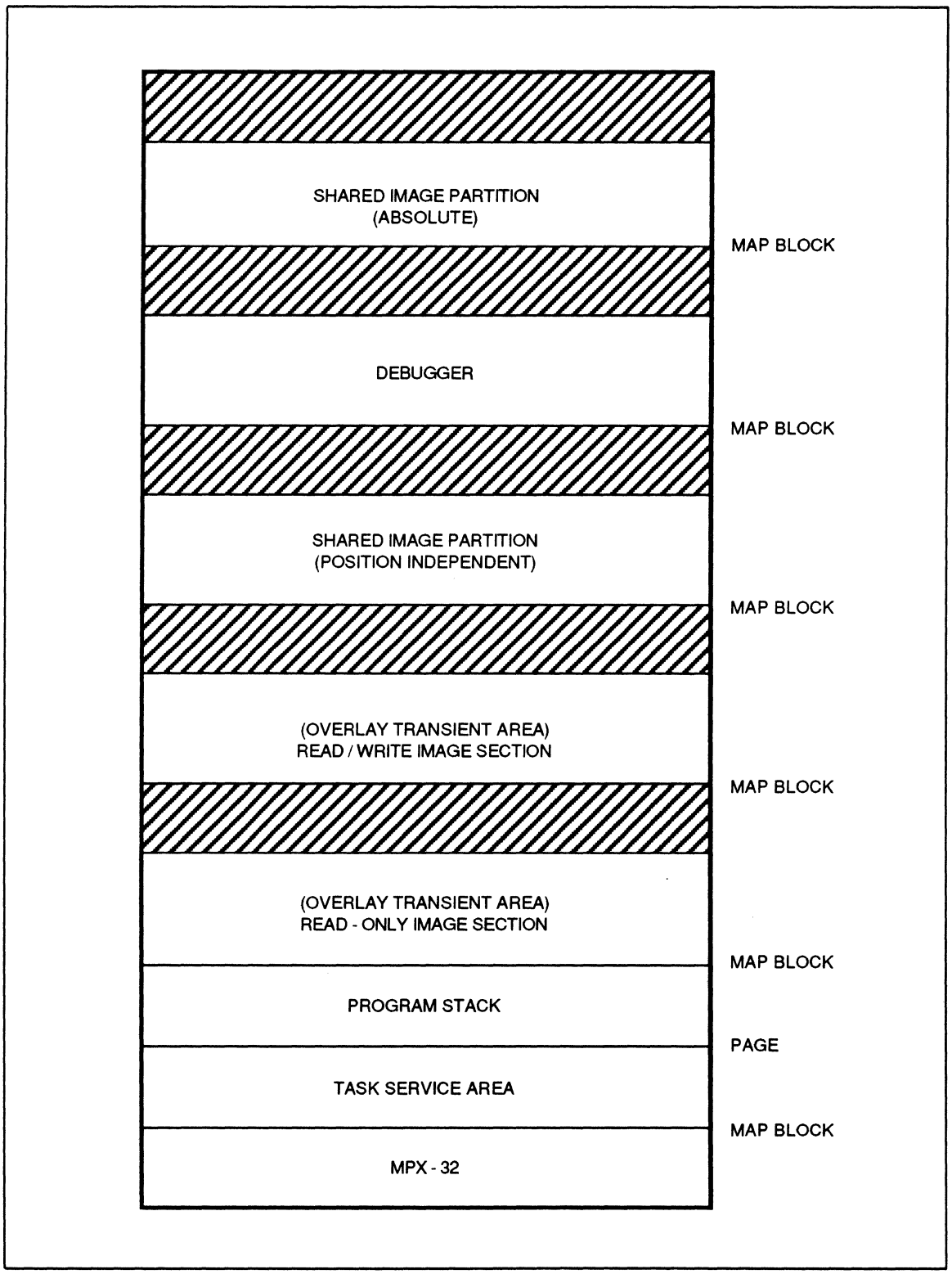


Figure 2-3
Base Mode Shared Task Address Space

2.3 Task Execution

Nonbase mode tasks are introduced to the system by a request to activate the cataloged load module by name. Activation can be requested in several different ways.

- Batch and interactive tasks are activated by the job control or TSM $RUN and $EXECUTE directives.
- Real-time tasks can be activated by the M.ACTV or M.PTSK system services. The requestor uses the M.PTSK service to rename the task or to specify additional or alternate resource requirements for the task.

  Real-time tasks can also be activated by the TSM $ACTIVATE directive the OPCOM ACTIVATE directive, timers, or interrupts.

The operating system enters base mode tasks in one of two ways: the initial dispatch of the task, or as a result of an asynchronous event which includes messages, breaks, or end-action notification. In either case, entry is through a call instruction.

2.3.1 Task Activation Sequencing (M.ACTV, M.PTSK)

The MPX-32 task management module performs task activation in two phases.

2.3.1.1 Phase 1 of Activation

When a task is activated by the M.ACTV service or the M.PTSK service, the MPX-32 resource manager runs for the task that issues the service call (the activating task). In many cases, the activating task is TSM or OPCOM. Running at the priority specified by the activating task, the resource manager constructs a rudimentary task service area (TSA) for the new task in the task's address space and a rudimentary dispatch queue entry (DQE) in the communications region. Data in the prototypes include: a task number, parameters passed with the task (M.PTSK), the load module information table (LMIT), and other basic data that define the task.

Initially, the DQE for the task is unlinked from the list of free DQE's maintained by the CPU scheduler and linked to the preactivation state queue (PREA). See the State Chain Management section in this chapter for information about the DQE. After the prototype TSA and DQE are constructed, the DQE is unlinked from the PREA state queue and linked to the appropriate ready-to-run queue. A context is set up in the prototype TSA so that the resource manager can gain control for the second phase of activation as soon as the new task becomes the highest priority ready-to-run task on the system. There are several cases where task activation does not continue at the end of phase 1:

- activation with a run request for a single-copied task that is already active
- timer activation requests (M.SETT)
- RTM-compatible activation on an interrupt (CALM X'66' or M.CONN)

In the first case, the CPU scheduler can link the run request to an existing DQE. See the User Run Receivers section in this chapter for information on task run receivers. In the last two cases, the task remains in the preactivation state queue until the timer expires or the interrupt fires. At that point, such tasks are linked to the appropriate ready-to-run queue as described previously.

2.3.1.2 Phase 2 of Activation

In this phase, the resource manager operates for the new task, and runs at the new task's specified priority. It reads in the resource requirements summary (RRS) from the load module file, merges them with static assignments, and validates the results. Resources are allocated. The task's DQE can be linked and unlinked to various state queues as it moves through stages of device and memory allocation.

If any parameters, assignments, or other task resource requirements specified in the load module or by job control or TSM assignments are invalid, the resource manager aborts the task during this phase and the task exits as described in the Task Termination Sequencing section of this chapter.

When the new task has allocated all resources required for execution, it is loaded into memory, relocated, and the resource manager transfers control to the task at its specified transfer address.

There are two exceptions to the control transfer at the end of phase 2. The first exception is a task that has been initiated by the OPCOM ESTABLISH directive. This task is linked into the suspended state queue (SUSP) instead of going into execution. This allows MPX-32, like the RTM, to activate a task that resides permanently in memory (a resident task). In MPX-32, resident means locked in memory. When an activating request occurs for a task that has been established (a timer expires, an interrupt is issued, or the task is resumed), the task is ready to execute and is brought into execution with just a context switch. If the task has been cataloged as resident, no inswap is required.

The second exception is a task that has been activated with the MPX-32 Debugger attached (TSM or job control DEBUG task name directive). Instead of transferring control to the task, the resource manager first loads and then transfers control to the debugger.

2.3.2 Task Service Area (TSA)

The task service area (TSA) is a section of memory associated with each active task. The size of each task's TSA is fixed for the duration of the task's execution. However, the sizes of TSA's among tasks is variable and is dependent on the task's logical address space size and the amount of space reserved for I/O activity.

As shown in Figure 2-4, the number of blocking buffers, file assignment table (FAT) entries, and the file pointer table (FPT) entries varies among tasks.

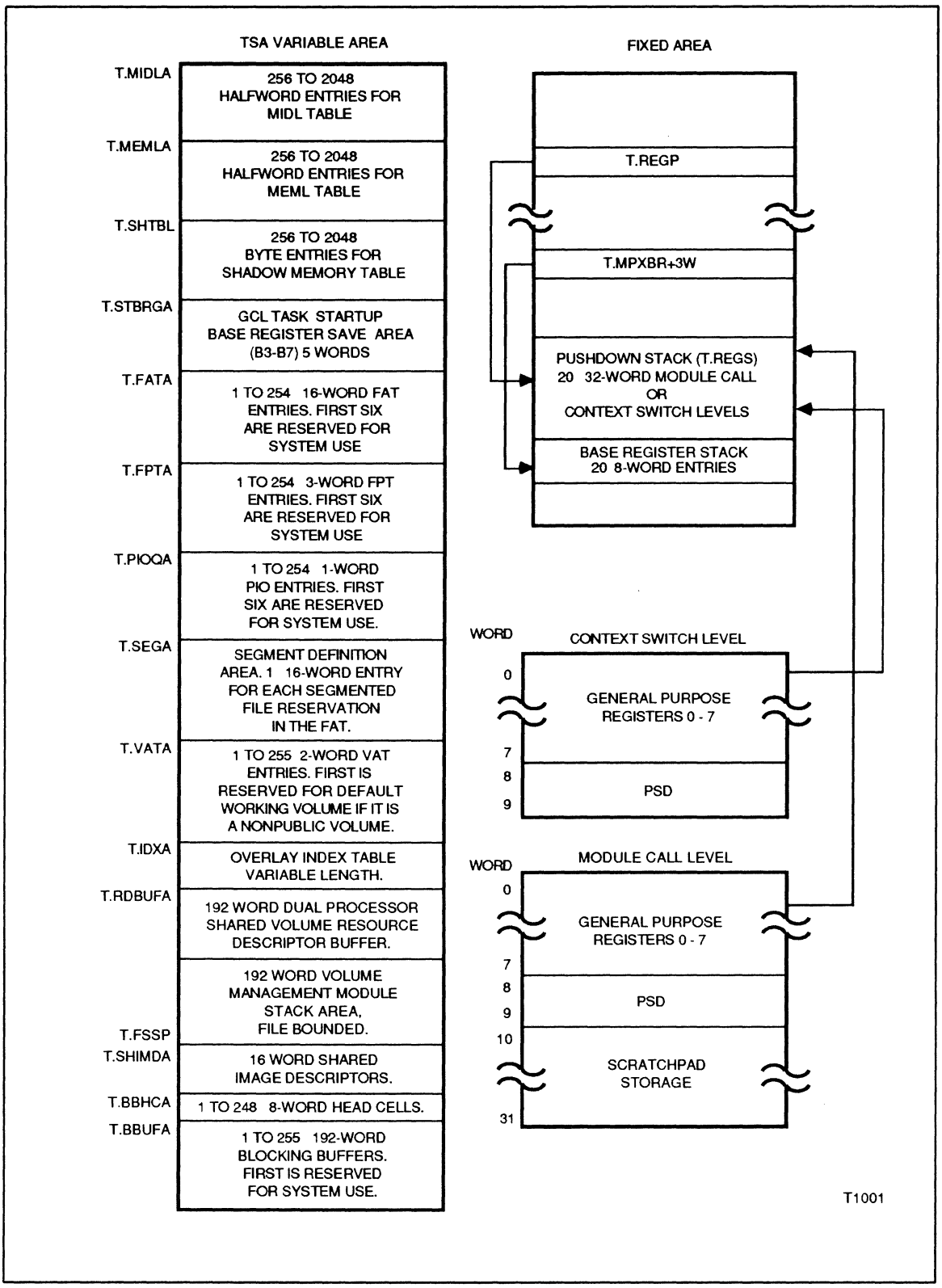


Figure 2-4
Task Service Area (TSA) Structure

For all tasks, a fixed number of buffers, FAT, and FPT entries are reserved for MPX-32 use. For example, they are present in every TSA.

The pushdown stack area in the TSA provides reentrancy in calls to system modules. At each call to a system module entry point, the stack pointer (T.REGP) is incremented to the next 32-word pushdown level where the contents of the general purpose registers (GPRs) and program status doubleword (PSD) are saved. Within this 32-word level, 22 words are available for scratchpad storage by the module entry point being called. T.REGP is decremented to the previous pushdown level upon return to the entry point caller. Upon context switch away from a task, the next pushdown level is used to preserve the contents of the task's registers and PSD. Ten words are used at the context switch level.

2.4 Central Processing Unit (CPU) Scheduling

The MPX-32 CPU scheduler is responsible for allocating CPU execution time to active tasks. Tasks are allocated CPU time based on execution priority and execution eligibility. Execution priority is specified when a task enters (is cataloged into) the system. Execution eligibility is determined by the task's readiness to run.

2.4.1 Execution Priorities

The MPX-32 system provides 64 levels of execution priority. These priority levels are divided into two major categories. Real-time tasks operate in the priority range 1 to 54. Time-distribution tasks operate in the priority range 55 to 64.

2.4.2 Real-Time Priority Levels (1 to 54)

MPX-32 schedules real-time tasks on a strict priority basis. The system does not impose time-slice, priority migration, or any other scheduling algorithm that interferes with the execution priority of a real-time task. Execution of an active real-time task at its specified priority level is inhibited only when it is ineligible for execution (not ready-to-run). Execution of a real-time task can always be preempted by a higher priority real-time task that is ready-to-run.

2.4.3 Time-Distribution Priority Levels (55 to 64)

For tasks executing at priority levels 55 to 64, MPX-32 provides a full range of priority migration, situational priority increment, and time-quantum control.

2.4.3.1 Priority Migration

The specified execution priority of a time-distribution task is used as the task's base execution priority. Each time-distribution task's current execution priority is determined by the base priority level as adjusted by any situational priority increment. The current execution priority is further adjusted by increasing the priority (by one level) whenever execution is preempted by a higher priority time-distribution task, and decreasing the priority whenever the task gains CPU control. The highest priority achievable by a time-distribution task is priority level 55. The lowest priority is clamped at the task's base execution level.

2.4.3.2 Situational Priority Increments

Time-distribution tasks are given situational priority increments to increase responsiveness. The effect of situational priority increments is to give execution preference to tasks that are ready-to-run after having been in a natural wait state. A task that is CPU bound migrates toward its base execution priority. Situational priority increments are invoked when a task is unlinked from a wait-state list, and relinked to the ready-to-run list.

Situation	Priority Increment
Terminal input wait complete	Base level + 2
I/O wait complete	Base level + 2
Message (send) wait complete	Base level + 2
Run request (send) complete	Base level + 2
Memory (inswap) wait complete	Base level + 3
Preempted by real-time task	Level 55

2.4.3.3 Time-Quantum Controls

MPX-32 allows for the specification of two time-quantum values at SYSGEN. If these values are not specified, system default values are used. The two quantum values are provided for scheduling control of time-distribution tasks. The first quantum value (stage 1) indicates the minimum amount of CPU execution time guaranteed to a task before preemption by a higher priority time-distribution task. The stage 1 quantum value is also used as a swap inhibit quantum after inswap. The second quantum value represents the task's full-time quantum. The difference between the first and second quantum values defines the execution period called quantum stage 2. During quantum stage 2, a task may be preempted and/or outswapped by any higher priority task. When a task's full-time quantum has expired, it is relinked to the bottom of the priority list, at its base execution priority.

Time-quantum accumulation is the accumulated sum of actual execution times used by this task. A task's quantum accumulation value is reset when the task voluntarily relinquishes CPU control, for example, suspends, performs wait I/O, etc.

2.4.4 State Chain Management

The current state of a task ready-to-run, waiting for I/O, etc., is reflected by the linkage of the dispatch queue entry (DQE) associated with the task into the appropriate state chain. Linkage is established via string forward and string backward addresses and a state queue index in each DQE. The string forward address for a given DQE points to the closest lower priority DQE and the string backward address points to the closest higher priority DQE in a given state. The index points to a state chain head cell, which contains the link forward/backward addresses from the DQE at the top (highest priority task) of the state chain. At a given time, from any one DQE or from a head cell, an entire state chain queue can be examined by moving either backward or forward through the DQE linkages.

The state queues are divided into two major categories: ready-to-run and waiting. See Table 2-2. The ready-to-run category is subdivided by priority, with a single queue for the real-time priorities and a separate queue for each of the time-distribution priority levels. The waiting category is subdivided according to the resource or event required to make the task eligible for execution.

Table 2-2
MPX-32 State Queues

State Index	Label	Meaning
0	FREE	DQE is available (in free list)
1	PREA	Task activation in progress
2	CURR	Task is executing
Ready-to-Run Queues		
3	SQRT	Task is ready-to run at priority level 1-54
4	SQ55	at priority level 55
5	SQ56	at priority level 56
6	SQ57	at priority level 57
7	SQ58	at priority level 58
8	SQ59	at priority level 59
9	SQ60	at priority level 60
10	SQ61	at priority level 61
11	SQ62	at priority level 62
12	SQ63	at priority level 63
13	SQ57	at priority level 64
Operation Wait Queues		
14	SWTI	Task is waiting for terminal input
15	SWIO	Task is waiting for I/O
16	SWSM	Task is waiting for message complete
17	SWSR	Task is waiting for run request complete
18	SWLO	Task is waiting for low speed output
Execution Wait Queues		
19	SUSP	Task is waiting for one of the following: . Timer expiration . Resume request . Message request interrupt

Continued on next page

Table 2-2
MPX-32 State Queues *(Continued)*

State Index	Label	Meaning
		Execution Wait Queues
20	RUNW	Task is waiting for one of the following . Timer expiration . Run request
21	HOLD	Task is waiting for a continue request
22	ANYW	Task is waiting for one of the following: . Timer expiration . No-wait I/O complete . No-wait message complete . No-wait run request complete . Message request interrupt . Break interrupt
		Resource Wait Queues
23	SWDC	Task is waiting for disc space allocation
24	SWDV	Task is waiting for device allocation
25	N/A	Reserved
26	MRQ	Task is waiting for memory allocation
27	SWMP	Task is waiting for memory pool allocation
28	SWGC	Task is waiting in general wait queue
		IPU State Queues*
29	CIPU	Current IPU task
30	RIPU	Requesting IPU task

* See the MPX-32 Technical Manual, Volume I for further details.

2.5 Internal Processing Unit (IPU)

The IPU is a user-transparent device managed by the MPX-32 operating system. The IPU is scheduled as an additional resource to offload the CPU and improve system throughput in a multitasking environment. The IPU can be used to execute task level code and a limited set of system services.

Scheduling of tasks for IPU execution is controlled by the CPU executive (H.EXEC) working with the IPU executive (H.CPU) for the standard scheduler. An optional scheduler uses a different CPU executive, H.EXEC2, and a different IPU executive, H.CPU2. The optional CPU/IPU scheduling logic is enabled by the SYSGEN DELTA directive. The standard scheduler is more processor oriented whereas the optional scheduler is more priority oriented. The following sections apply to both schedulers; any differences are noted.

2.5.1 Options

Options for the IPU can be specified at catalog or execution time by TSM. The IPU options are:

- IPUBIAS — When set, tasks that are IPU eligible are run by the IPU. Any time during execution where eligibility ceases, the CPU is trapped and the task is scheduled to execute at its cataloged priority in the CPU.
- CPUONLY — When set, the IPU is ignored and the task is executed by the CPU.

If not specified, the first eligible processor executes the task.

Tasks that are compute bound may be biased to the IPU; tasks that are I/O bound may be designated to run only in the CPU.

The CPU/IPU scheduling logic automatically adapts to tasks that alternate between bursts of computing and bursts of I/O for nonbiased tasks.

2.5.2 Biased Task Prioritization

2.5.2.1 Standard CPU/IPU Scheduler

If the IPU scheduler finds more than one IPU-biased task waiting for processing, it places the tasks in a ready-to-run queue (C.RIPU) in priority order among themselves. The tasks are eligible for swapping while waiting.

2.5.2.2 Optional CPU/IPU Scheduler

Tasks that are IPU biased are not enqueued on the IPU ready to run queue (C.RIPU). These tasks are linked to the ready to run lists SQRT through SQ64 with other task types. Since the IPU-biased tasks do not enter a wait state, they are less likely candidates for swapping than tasks that are in the wait state.

IPU-biased tasks may have their priority boosted using the SYSGEN DELTA directive. If the DELTA directive is set to 0, scheduling occurs on a priority basis only. If the DELTA value is greater than 0 and less than or equal to 54, the value is subtracted from the cataloged priority (boosting its priority) at scheduling time. For example, when the DELTA is set to 5, a priority 20 IPU biased task competes for the IPU at priority 15. Similarly, when an IPU bias task needs the CPU for a system service, the boosted priority (15) is used to compete for the CPU. The DELTA does not apply when an IPU-biased task executes task level code in the CPU.

2.5.3 Nonbiased Task Prioritization

If the IPU scheduler finds more than one nonbiased task waiting for processing (any task in ready state queues SQRT thru SQ64), it places them in priority order among themselves and schedules them for processing after execution. The highest priority IPU-eligible task is scheduled in the IPU regardless of its bias or unbiased attribute.

2.5.4 IPU Task Selection and Execution

When the IPU task scheduler has found a task, it checks for IPU eligibility. For a task to be eligible for IPU execution, the following conditions must be present:

- no pending task interrupts
- no system action requests, for example, aborts
- not CPU biased
- current execution address outside of resident operating system

If a task fails any one of these tests, it is ineligible for IPU execution (i.e., ignored) and the task scheduler proceeds to select the next task, if any.

If a task has been selected and is determined eligible for IPU processing, it is linked to the current IPU task queue (C.CIPU), a Start IPU (SIPU) is executed from the CPU, the IPU executive (H.IPU) fields the trap, loads the task's map registers (LPSDCM), and executes the task.

Tasks running with batch priorities (55-64) are not subject to time distribution while being executed in the IPU.

Note: Tasks running with batch priorities (55-64) cannot have their priorities boosted via the DELTA value.

2.5.5 CPU Execution of IPU Tasks

2.5.5.1 Standard CPU/IPU Scheduler

Unbiased tasks require CPU execution for code sequences requiring operating system execution. Unbiased tasks are also free to execute task level code in the CPU.

IPU biased tasks are executed by the CPU for only those code sequences requiring operating system execution. When the PSD points back into the task, its CPU execution is terminated immediately and the task is linked to the IPU request queue (C.RIPU). If the IPU is running and this new task has a higher priority (lower number) than the task the IPU is executing, the executing task is preempted by the new task and replaced by the higher priority task. If the IPU is running and the new task has a lower priority (higher number) than the task currently under execution, the new task is placed in the IPU ready-to-run queue (C.RIPU).

2.5.5.2 Optional CPU/IPU Scheduler

When the highest and second highest priority tasks are IPU biased, the CPU executes task level code of the second highest priority task. However, the task's priority will not be boosted by the DELTA value in this case.

2.5.6 Priority versus Biasing

When there is a task in the IPU and it encounters a code sequence requiring CPU execution, the task is linked to a ready-to-run state chain at its base priority.

Note: For the optional CPU/IPU scheduler, an IPU bias task is linked to the ready to run state at base priority minus the DELTA value for code sequences requiring CPU execution.

An IPU task that requires some CPU execution cannot execute in the CPU if a CPU-only task of the same priority is in the CPU.

An IPU task that requires some CPU execution can execute in the CPU if:

- a non-CPU-only task of the same priority is in the CPU.
- the task in the CPU has a lower priority.
- there is no task in the CPU.

2.5.7 IPU Accounting

When the IPU and its interval timer handler are specified during SYSGEN, and the IPU is used for task execution, the following message displays at EOJ and when logging off a terminal:

```
IPU EXECUTION TIME = xx HOURS- xx MINUTES- xx.xx SECONDS
```

xx is a decimal number

The IPU accounting can be turned off to reduce context switch time. Refer to the Real-time Accounting On/Off section of this chapter for more information.

2.5.8 IPU Executable System Services

When the execution address of the task is within the resident operating system, the task cannot be scheduled to be executed by the IPU. However, when the execution address of the task is within the task, the task can be executed by the IPU. Once the task is in the IPU, the IPU can execute a limited set of system services. These are memory reference only system services, since the IPU cannot execute any I/O instructions. The system services that are executable in the IPU are listed in the Nonbase Mode and Base Mode System Services chapters of this volume.

2.5.9 IPU Scheduling

Although the IPU is scheduled transparently by the operating system, users can restrict a task to execute only in the CPU or bias a task to execute in the IPU. Tasks designated as CPU only cannot execute in the IPU. IPU biased tasks and unbiased tasks must meet the following requirements to execute in the IPU:

- no pending or active task interrupts. For example, I/O end action, breaks, etc. A pending interrupt is an interrupt that has been recognized by the operating system but has not yet been dispatched to the task.
- no system action requests; for example, abort, hold, etc.
- no context switching inhibited
- no temporary inhibit from IPU execution because it contains instructions not executable by the IPU; for example, CD, BEI, etc.
- execution address of the task must be outside the resident operating system
- starting logical address of the task's task service area (TSA) must be greater than the logical end of the resident operating system (i.e., C.LOSEND). All user tasks meet this requirement. System resident modules and tasks do not meet this requirement.

If a task does not meet the above requirements, it cannot be executed in the IPU and is scheduled for execution in the CPU.

Scheduling unbiased tasks includes checks for the following conditions:

- task is currently executing in the CPU
- task currently executing in the CPU that is not eligible to execute in the IPU
- is a task currently executing in the CPU that is eligible to execute in the IPU

If a task is not executing in the CPU, S.EXEC20, the main scheduling routine, attempts to schedule a task for the IPU. If IPU eligible tasks are found, the task with the highest priority is scheduled for IPU execution. S.EXEC20 then schedules the highest priority ready-to-run task for CPU execution. IPU eligible tasks are automatically considered CPU eligible. If an eligible task for either processor cannot be found, that processor remains idle.

Note: For the optional CPU/IPU scheduler, tasks that are IPU biased may run on the CPU.

If a task is currently executing in the CPU and it is not eligible for IPU execution, it continues to be executed by the CPU. S.EXEC20 attempts to schedule a task for the IPU. If an IPU eligible task cannot be found, the IPU remains idle.

If a task is currently executing in the CPU and is IPU eligible, the following factors are considered in the order described by S.EXEC20:

- If the current task is IPU biased and the IPU is idle, the task is scheduled for IPU execution.
- If the current task is IPU biased and the IPU is executing a task with a higher priority than the current task, the current task is placed in the IPU request state (RIPU). If the current task has a higher priority than the task executing in the IPU, the task executing in the IPU is removed from execution and the current IPU biased task is scheduled for execution.

  For the optional CPU/IPU scheduler, if the current task is IPU biased, but the IPU is executing a higher priority task, the CPU will run the current task.
- If the IPU is idle, S.EXEC20 performs a check to see if another task is requesting CPU execution. If no other task is found, the current task remains in execution in the CPU. If another task is found, the current CPU task is moved to the IPU for execution. The highest priority task of the other tasks found is scheduled for CPU execution.

  For the optional CPU/IPU scheduler, if the IPU is idle and the current task is IPU eligible, the task is scheduled for the IPU.
- If the IPU is busy, S.EXEC20 performs a check to see if another task is requesting CPU execution. If no other task is found, the current task remains in execution in the CPU. If a nonreal-time task is found, the current task remains in the CPU. If a real-time task is found, the priority of the current task executing in the CPU is compared with the priority of the current task executing in the IPU. If the CPU task has a higher priority, the task in the IPU is replaced by the task in the CPU. Otherwise, the current task remains in execution in the CPU.

These additional factors, considered in IPU scheduling, allow for a more predictable operation and eliminate unnecessary scheduling overhead. Unless the user can be assured of benefits through the use of IPU biasing or CPU-only restrictions, it is recommended that tasks be run unbiased. This allows the MPX-32 executive to make the decision on IPU usage.

2.6 Scheduling Task Interrupts

In addition to the 64 execution priority levels available, the MPX-32 scheduler provides a software interrupt facility within the individual task environment.

2.6.1 Task Interrupt Levels

Individual tasks operating in the MPX-32 environment may be organized to take advantage of task unique software interrupt levels. Each task in the MPX-32 system can have six levels of software interrupt, sometimes referred to as pseudo-interrupts:

Level Priority	Description
0	reserved for operating system use
1	debug
2	break
3	end action
4	message
5	normal execution — run request

2.6.1.1 Task Interrupt Receivers

An individual task is allowed to issue system service calls to establish interrupt receiver addresses for both break and message interrupts. The debugger interrupt level is used by the system to process tasks running in debug mode. The end action interrupt level is used for system postprocessing of no-wait I/O, message, or run requests. It is also used for executing end action routines specified by the user task. The normal execution level is used for run-request processing and general base level task execution.

2.6.1.2 Scheduling

Task interrupt processing is gated by the CPU scheduler during system service processing. If a task interrupt request occurs while the task is executing in a system service, the scheduler defers the interrupt until the service returns to the user task execution area. If service calls are nested, the scheduler defers the task interrupt until the last service executes and returns to the user task execution area. The user can defer task interrupts through calls to Synchronize Task Interrupts (M.SYNCH) or Disable Message Interrupts (M.DSMI).

2.6.1.3 System Service Calls from Task Interrupt Levels

A task can use the complete set of system services from any task interrupt level. Tasks are prohibited from making Wait-For-Any calls (M.ANYW, M.EAWAIT) from task interrupt levels.

2.6.1.4 Task Interrupt Context Storage

When a task interrupt occurs, the CPU scheduler automatically stores the interrupted context into the TSA pushdown stack. This context is automatically restored when the task exits from the active interrupt level.

2.6.1.5 Task Interrupt Level Gating

When a task interrupt occurs, the level is marked active. Additional interrupt requests for that level are queued until the level active status is reset by the appropriate system service call. When the level active status is reset, any queued request is processed.

2.6.2 User Break Interrupt Receivers (M.BRK, M.BRKXIT)

A task enables the break interrupt level by calling the M.BRK service to establish a break interrupt receiver address. The level becomes active as a result of a break interrupt request generated either from a hardware break or from an M.INT service call which specified this task. When the break level is active, end action, message, and normal execution processing are inhibited. The level active status is reset by calling the M.BRKXIT service to exit from the pseudo-interrupt (break) level.

2.7 Intertask Communication

MPX-32 provides both message request and run-request send/receive processing. Run-request services allow a task to queue an execution request with optional parameter passing for another task. Message services allow a task to send a message to another active task. The services provided for use by the destination tasks are called receiving task services. Those provided for tasks which issue the requests are called sending task services. Message and run-request services use the software interrupt scheduling structure described in the previous section, Scheduling Task Interrupts.

2.7.1 User End-Action Receivers (M.XMEA, M.XREA, M.XIEA)

When a task issues a no-wait I/O, a message request, or a run request, a user-task end-action routine address can be specified. If specified, the routine is entered at the end-action priority level from the appropriate system postprocessing routine. When the end-action level is active, processing at the message or normal execution level is inhibited. The level active status is reset by calling the appropriate end-action service:

End-Action Type	End-Action Exit Service
I/O	SVC 1,X'2C'
Send message	M.XMEA
Send run request	M.XREA

2.7.2 User Message Receivers (M.RCVR, M.GMSGP, M.XMSGR)

A task can enable the message interrupt level by calling the M.RCVR system service to establish a message interrupt receiver address. The level becomes active as the result of a message send request specifying this task as the destination task.

When the message level is active, normal execution processing is inhibited. The task's receiver can call the M.GMSGP system service to store the message in a user receiver buffer. After appropriate processing, the message interrupt level may be reset by calling the M.XMSGR system service to exit from the message interrupt receiver.

2.7.3 User Run Receivers (M.GRUNP, M.XRUNR)

User run receivers execute at the normal task execution base level. The cataloged transfer address is used as the run-receiver execution address. The run-receiver mechanism is provided by the system to allow queued requests for task execution with optional parameter passing.

When a run request is issued by the M.SRUNR service, the task load module name may be used to identify the task to be executed. If a task of that load module name is currently active and single-copied, the run request is queued from its existing DQE. If the specified task is not active, or if the task is not a single-copied task, it is activated and the run request is then linked to the new DQE. A new copy is activated for each run request sent to a multicopied task by load module name or pathname vector. If the multicopied task is waiting for a run request, for example, in the RUNW state chain, the task number must be specified.

The task receiving the run request can call the M.GRUNP system service to store the run parameters in a user receiver buffer. After appropriate processing, the run-receiver task can exit by calling the M.XRUNR system service. Any queued run requests are then processed.

When a task in the run-receiver mode enters its abort receiver, the run request has already been terminated and the task issuing the run request has already received status or call back depending on the options used. A new copy of the task is activated to satisfy any queued run requests.

2.7.4 Receiving Task Services

2.7.4.1 Establishing Message Receivers (M.RCVR)

To receive messages sent from other tasks, a task must be active and have a message receiver established. A message receiver is established by calling the system service M.RCVR, and providing the receiver routine address as an argument with the call.

2.7.4.2 Establishing Run Receivers

Any valid task can be a run receiver. Although a set of special run receiver services are provided, in the most simple case, they need not be used. The run receiver mechanism is provided by the system to allow queued requests for task execution, with optional parameter passing. The cataloged transfer address is used as the run receiver execution address. The task load module name is used to identify the task to be executed. If a run request is issued for a task not currently active, the task is activated automatically. If the task is single-copied and currently active, the run request is queued until the task exits. If the task is multicopied and currently active, the load module is activated (multicopied) to process this request. When a single-copied task exits, any queued run requests are executed.

2.7.4.3 Executing Message Receiver Programs

When a task is active and has an established message receiver, it can receive messages sent from other tasks. A message sent to this task causes a software (task) interrupt entry to the established message receiver.

2.7.4.4 Executing Run Receiver Programs

When a valid task is executed as a result of a run request sent by another task, it is entered at its cataloged transfer address. A run receiver executes at the normal task execution (base) level.

2.7.4.5 Obtaining Message Parameters (M.GMSGP)

When the message receiver is entered, R1 contains the address of the message queue entry in memory pool. The task can retrieve the message directly from memory pool, or the task can call a receiver service (M.GMSGP) to store the message into the designated receiver buffer. If the M.GMSGP service is used, the task must present the address of a 5-word parameter receive block (PRB) as an argument with the call.

2.7.4.6 Obtaining Run Request Parameters (M.GRUNP)

When the run receiver is entered, R1 contains the address of the run-request queue entry in memory pool. The task can retrieve the run request parameters directly from memory pool, or the task can call a receiver service (M.GRUNP) to store the run request parameters into the designated receiver buffer. If the M.GRUNP service is used, the task must present the address of a 5-word parameter receive block (PRB) as an argument with the call.

2.7.4.7 Exiting the Message Receiver (M.XMSGR)

When processing of the message is complete, the message interrupt level must be exited by calling the M.XMSGR service. When M.XMSGR is called, the address of a two-word receiver exit block (RXB) must be provided. The RXB contains the address of the return parameter buffer, and the number of bytes (if any) to be returned to the sending task. The RXB also contains a return status byte to be stored in the parameter send block (PSB) of the sending task. After message exit processing is complete, the message receiver queue for this task is examined for any additional messages to process. If none exists, a return to the base level interrupted context is performed.

2.7.4.8 Exiting the Run Receiver Task (M.EXIT, M.XRUNR)

When run-request processing is complete, the task can use either the standard exit call (M.EXIT), or the special run-receiver exit service (M.XRUNR).

If the standard exit service (M.EXIT) exits the run-receiver task, no user status or parameters are returned. Only completion status is posted in the scheduler status word of the parameter send block (PSB) in the sending task. After completion processing for the run request is accomplished, the run receiver queue for this task is examined, and any queued run request causes the task to be re-executed. If the run-receiver queue for this task is empty, a standard exit is performed.

If the special exit (M.XRUNR) exits the run-receiver task, the address of a 2-word receiver exit block (RXB) must be provided as an argument with the call. The RXB contains the address of the return parameter buffer, and the number of bytes (if any) to be returned to the sending task. The RXB also contains a return status byte to be stored in the parameter send block (PSB) of the sending task. After completion processing for the run request is accomplished, the exit control options in the RXB are examined. If the wait exit option is used, the run receiver queue for this task is examined for any additional run requests to be processed. If none exist, the task is put into a wait-state, waiting for the receipt of new run requests. Execution of the task does not resume until such a request is received. If the terminate exit option is used, any queued run requests are processed. If the run receiver is empty, however, a standard exit is performed.

2.7.4.9 Waiting for the Next Request (M.SUSP, M.ANYW, M.EAWAIT)

In addition to the wait options described in the Exiting the Run Receiver Task section, a task can use the M.SUSP, M.ANYW, or M.EAWAIT system service. When operating at the base execution level, a task that has established a message receiver can use the M.SUSP service call to enter a wait-state until the next message is received.

A task may also make use of the special M.ANYW service from the base software level. The M.ANYW service is similar to M.SUSP. The difference is that whereas the M.SUSP wait-state is ended only upon receipt of a message interrupt, timer expiration, or resume, the M.ANYW wait-state is ended upon receipt of any message, end action, or break software interrupt.

M.EAWAIT is similar to M.ANYW except that if no requests are outstanding, an immediate return is made to the caller.

2.7.5 Sending Task Services

2.7.5.1 Message Send Service (M.SMSGR)

A task can send a message to another active task, providing the destination task has established a message receiver. The sending task must identify the destination task by task number. When the send message service (M.SMSGR) is called, the word bounded address of a PSB must be provided as an argument. The PSB specifies the message to be sent, whether or not any parameters are to be returned, and the address of a user end-action routine. User status can be returned by the destination task. The operating system also returns completion status in the PSB. No-wait and no-call-back control options are also provided. An unprivileged user is limited to five no-wait messages or to the value specified by the SYSGEN parameter MMSG.

2.7.5.2 Send Run-Request Service (M.SRUNR)

A task can send a run request to any active or inactive task, identifying the task by load module name. When the run request service (M.SRUNR) is called, the word bounded address of a PSB must be provided as an argument. The PSB format allows for the specification of the run request parameters to be sent, any parameters to be returned, scheduler and user status, as well as the address of a user end-action routine. No-wait and no-call-back control options are also provided. An unprivileged user is limited to five no-wait run requests.

Note: If a task activated with the TSM $ACTIVATE directive is sent a run request, the queued run request is ignored. However, if the task is activated with a run request and a second run request is sent to it, the queued run request is executed.

2.7.5.3 Waiting for Message Completion

A message can be sent in the wait or no-wait mode. If the wait mode is used, execution of the sending task is deferred until processing of the message by the destination task is complete. If the no-wait mode is used, execution of the sending task continues as soon as the request has been queued. The operation in progress bit in the scheduler status field of the PSB may be examined to determine completion. A sending task can issue a series of no-wait mode messages followed by a call to the M.ANYW or M.EAWAIT system wait service. This allows a task to wait for the completion of any no-wait messages previously sent. The completion of such a message causes resumption at the point after the M.ANYW or M.EAWAIT call.

2.7.5.4 Waiting for Run-Request Completion

Waiting for a run-request completion follows the same form and has the same options as waiting for message completion.

2.7.5.5 Message End-Action Processing (M.XMEA)

User specified end-action routines associated with no-wait message send requests are entered at the end-action software interrupt level when the requested message processing is complete. Status and return parameters are posted as appropriate. When end-action processing is complete, the M.XMEA service must be called to exit the end-action software interrupt level.

2.7.5.6 Run-Request End-Action Processing (M.XREA)

Run-request end-action processing follows the same form and has the same options as message end-action processing. The only difference is that the M.XREA service is used instead of M.XMEA.

2.7.6 Parameter Blocks

Parameters for run requests and messages are passed by parameter blocks established within the user task. The parameter blocks are described in this section.

2.7.6.1 Parameter Send Block (PSB)

The PSB describes a send request issued from one task to another. The same PSB format is used for both message and run requests. The address of the PSB (word bounded) must be specified when invoking the M.SMSGR or M.SRUNR services, but is optional when invoking the M.PTSK service.

When a load module name is supplied in words 0 and 1 of the PSB, the operating system searches the system directory only. For activations in directories other than the system directory, a pathname or RID vector must be supplied.

When activating a task with the M.SRUNR or M.PTSK service, the value specified in byte 0 of PSB word 2 (PSB.PRI) is used to determine the task's execution priority. This value overrides the cataloged priorities of the sending and receiving tasks and the priority specified in the PTASK parameter block. However, priority clamping is used to prevent time-distribution tasks from using this value to execute at a real-time priority, and real-time tasks from executing at a time-distribution priority. Values that can be specified in PSB.PRI are 1-64 (to be the task priority), 0 (to use the base priority of the sending task), and X'FF' (to ignore the PSB priority field).

A PSB can be specified as a parameter for the M.PTSK service, along with the required task activation (PTASK) block. The PTASK block also contains a priority specification field. The PSB priority value always overrides the PTASK block priority value.

A task number, not a load module name, must be used if sending a message request or if sending a run request to a multicopied task which is waiting for a run request.

	0 7	8 15	16 23	24 31
Word 0	Load module name (or task number if message or run request to multicopied task). See Note 1.			
1	Load module name or pathname vector or RID vector if activation (or 0 if message or run request to multicopied task). See Note 2.			
2	Priority (PSB.PRI). See Note 3.	Reserved	Number of bytes to be sent PSB.SQUA). See Note 4.	
3	Reserved	Send buffer address (PSB.SBA). See Note 5.		
4	Return parameter buffer length (bytes) (PSB.RPBL). See Note 6.		Number of bytes returned (PSB.ACRP). See Note 7.	
5	Reserved	Return parameter buffer address (PSB.RBA). See Note 8.		
6	Reserved	No-wait request end-action address (PSB.EAA). See Note 9.		
7	Completion status (PSB.CST). See Note 10.	Processing start status (PSB.IST). See Note 11.	User status (PSB.UST). See Note 12.	Options (PSB.OPT). See Note 13.

Notes:

1. Word 0, bits 0-31:

   For send message: Task number of the task to receive the message.

   For run request: Zero if using pathname vector or RID vector in word 1, else task number (word 1 must be 0), else characters 1 to 4 of the name of the load module to receive the run request.

2. Word 1, bits 0-31:

   For send message: Zero.

   For run request: Zero if using task number in word 0, else pathname vector or rid vector (word 0 must be zero), else characters 5-8 of the load module to receive the run request.

3. Word 2, bits 0-7: Priority (PSB.PRI) — contains the priority at which the receiver task is expected to be activated. Valid values are 1-64: 0 signifies the base priority of the sending task and X'FF' generates activation priority based on a combination of values that can be specified during task activation. The following tables show how the priority of a receiver task is determined when activated with M.SRUNR or with M.PTSK.

When Activating with M.SRUNR

Send Task	Cataloged Priority of Receive Task	Priority in PSB	Activates Receive Task at
1-54	1-54	0	Send Task Cataloged Priority
1-54	55-64	0	55 (Time-Distributed Clamp)
55-64	1-54	0	54 (Real-Time Clamp)
55-64	55-64	0	Send Task Cataloged Priority
*	1-54	1-54	PSB Priority
*	1-54	55-64	54 (Real-Time Clamp)
*	55-64	1-54	55 (Time-Distributed Clamp)
*	55-64	55-64	PSB Priority
*	*	X'FF'	Receive Task Cataloged Priority

* none specified

When Activating with M.PTSK

Send Task	Cataloged Priority of Receive Task	Priority in PTASK Block	PSB	Activates Receive task at
1-54	1-54	0	0	Send Task Cataloged Priority
1-54	55-64	0	0	55 (Time-Distributed Clamp)
1-54	*	1-54	0	Send Task Cataloged Priority
1-54	*	55-64	0	55 (Time-Distributed Clamp)
55-64	1-54	0	0	54 (Real-Time Clamp)
55-64	55-64	0	0	Send Task Cataloged Priority
55-64	*	1-54	0	54 (Real-Time Clamp)
55-64	*	55-64	0	Send Task Catalog Priority
*	1-54	0	1-54	PSB Priority
*	1-54	0	55-64	54 (Real-Time Clamp)
*	55-64	0	1-54	55 (Time-Distributed Clamp)
*	55-64	0	55-64	PSB Priority
*	*	1-54	1-54	PSB Priority
*	*	1-54	55-64	54 (Real-Time Clamp)
*	*	1-54	X'FF'	PTASK Block Priority
*	*	55-64	1-54	55 (Real-Time Clamp)
*	*	55-64	55-64	PSB Priority
*	*	55-64	X'FF'	PTASK Block Priority
*	*	0	X'FF'	Receive Task Cataloged Priority

* none specified

4. Word 2, bits 16-31: Number of bytes to be sent (PSB.SQUA) — specifies the number of bytes to be passed (0 to 768) with the message or run request.
5. Word 3, bits 8-31: Send buffer address (PSB.SBA) — contains the word address of the buffer containing the parameters to be sent.
6. Word 4, bits 0-15: Return parameter buffer length (PSB.RPBL) contains the maximum number of bytes (0-768) that may be accepted as returned parameters.
7. Word 4, bits 16-31: Number of bytes actually returned (PSB.ACRP) is set by the send message or run request service upon completion of the request.
8. Word 5, bits 8-31: Return parameter buffer address (PSB.RBA) contains the word address of the buffer into which any returned parameters are stored.
9. Word 6, bits 8-31: No-wait request end-action address (PSB.EAA) contains the address of a user routine to be executed at an interrupt level upon completion of the request.
10. Word 7, bits 0-7: Completion status (PSB.CST) is a bit encoded field that contains completion status information posted by the operating system as follows:

Bit	Meaning When Set
0	operation in progress (busy)
1	destination task was aborted before completion of processing for this request
2	destination task was deleted before completion of processing for this request
3	return parameters truncated (attempted return exceeds return parameter buffer length)
4	send parameters truncated (attempted send exceeds destination task receiver buffer length)
5	rser end action routine not executed because of task abort outstanding for this task (may be examined in abort receiver to determine incomplete operation)
6-7	reserved

11. Word 7, bits 8-15: Processing start (initial) status (PSB.IST) is a value encoded field that contains initial status information posted by the operating system as follows:

Code	Definition
0	normal initial status
1	message request task number invalid
2	run request load module name not found in directory
3	reserved
4	file associated with run request load module name oes not have a valid load module format
5	dispatch queue entry (DQE) space is unavailable for activation of the load module specified by a run request
6	an I/O error was encountered while reading the directory to obtain the file definition of the load module specified in a run request
7	an I/O error was encountered while reading the file containing the load module specified in a run request.
8	memory unavailable
9	invalid task number for run request to multicopied load module in RUNW state
10	invalid priority specification — an unprivileged task cannot specify a priority which is higher than its own execution priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task receiver queue is full

12. Word 7, bits 16-23: User Status (PSB.UST) — As defined by sending and receiving tasks.
13. Word 7, bits 24-31: Options (PSB.OPT) contains user request control specification. It is bit encoded as follows:

Bit	Meaning When Set
24	request is to be issued in no-wait mode
25	do not post completion status or accept return parameters. This bit is examined only if bit 24 is set. When this bit is set, the request was issued in the no call-back mode.

2.7.6.2 Parameter Receive Block (PRB)

The PRB is used to control the storage of passed parameters into the receiver buffer of the destination task. The same format PRB is used for both message and run requests. The address of the PRB must be presented when either the M.GMSGP or M.GRUNP services are invoked by the receiving task.

	0 7	8 15	16 23	24 31
Word 0	Status (PRB.ST). See Note 1.	Parameter receiver buffer address (PRB.RBA). See Note 2.		
1	Receiver buffer length (bytes) (PRB.RBL). See Note 3.		Number of bytes received (PRB.ARQ). See Note 4.	
2	Owner name of sending task (Word 1) (PRB.OWN). See Note 5.			
3	Owner name of sending task (Word 2) (PRB.OWN). See Note 5.			
4	Task number of sending task (PRB.TSKN). See Note 6.			

Notes:

1. Status (PRS.ST) contains the status-value encoded status byte:

Code	Definition
0	normal status
1	invalid PRB address (PRB.ER01)
2	invalid receiver buffer address or size detected during parameter validation (PRB.RBAE)
3	no active send request (PRB.NSRE)
4	receiver buffer length exceeded (PRB.RBLE)

2. Parameter receiver buffer address (PRB.RBA) contains the word address of the buffer where the sent parameters are stored.
3. Receiver buffer length (PRB.RBL) contains the length of the receiver buffer (0 to 768 bytes).
4. Number of bytes received (PRB.ARQ) is set by the operating system and is clamped to a maximum equal to the receiver buffer length.
5. Owner name of sending task (PRB.OWN) is a doubleword that is set by the operating system to contain the owner name of the task that issued the parameter send request.
6. Task number of sending task (PRB.TSKN) is set by the operating system to contain the task activation sequence number of the task that issued the parameter send request.

2.7.6.3 Receiver Exit Block (RXB)

The receiver exit block (RXB) controls the return of parameters and status from the destination (receiving) task to the task that issued the send request. It is also used to specify receiver exit-type options. The same format RXB is used for both messages and run requests. The address of the RXB must be presented as an argument when either the M.XMSGR or M.XRUNR services are called.

	0 7	8 15	16 23	24 31
Word 0	Return status (RXB.ST). See Note 1.	Return parameter buffer address (RXB.RBA). See Note 2.		
1	Options (RXB.OPT). See Note 3.	Reserved	Number of bytes to be returned (RXB.RQ). See Note 4.	

Notes:

1. Return status (RXB.ST) contains status as defined by the receiver task. Used to set the user status byte in the parameter send block (PSB) of the task which issued the send request.
2. Return parameter buffer address (RXB.RBA) contains the word address of the buffer containing the parameters which are to be returned to the task which issued the send request.
3. Options (RXB.OPT) contains receiver exit control options. It is encoded as follows:

Value	Exit Type	Meaning
0	M.XRUNR	wait for next run request.
	M.XMSGR	return to point of task interrupt.
1	M.XRUNR	exit task, process any additional run requests. If none exist, perform a standard exit.
	M.XMSGR	N/A

4. Number of bytes to be returned (RXB.PQ) contains the number of bytes (0 to 768) of information to be returned to the sending task.

2.7.7 User Abort Receivers (M.SUAR)

User abort receivers execute at the normal task execution base level. The user task can establish an abort receiver by calling the M.SUAR service.

If an abort condition is encountered during task operation, control is transferred to the task's abort receiver. Before entry, any active software interrupt level is reset, all outstanding operations or resource waits are completed, and all no-wait requests are processed. End-action routines associated with no-wait requests that complete while the abort is outstanding are not executed. Status bits reflecting this are posted in the appropriate FCBs. Any files opened or resources allocated at the time the abort condition is encountered remain opened and/or allocated when the abort receiver is executed.

The TSA stack is clean. The context at the time the abort condition is encountered is stored in T.CONTXT. When the abort receiver is entered, R5 reflects task interrupt status when the abort condition was encountered.

Bit	Meaning if Set
0-18	N/A
19	user break interrupt active.
20	end-action interrupt active.
21	message interrupt active.
22-31	N/A

The standard exit service (described in the Task Termination Sequencing section of this chapter) exits from a task's abort receiver. If another abort condition is encountered while a task is executing an abort receiver, the task is deleted.

A privileged task can reestablish its abort receiver through the M.SUAR service. An unprivileged task is not allowed to reestablish its abort receiver after an abort condition has been encountered. An attempt to do so results in a task delete.

2.7.8 Task Interrupt Services Summary

Table 2-3 summarizes the services described in this section including required parameter blocks. For a detailed description of the parameter blocks for run and message requests, see the Parameter Blocks section of this chapter.

2.7.9 Arithmetic Exception Handling

MPX-32 maintains a trap handler with the capability to do special handling of arithmetic exceptions generated by a task. If the arithmetic exception trap is enabled, any task can test for the occurrence of an exception via the T.EXCP flag in the T.BIT1 field of the TSA. For certain instructions, the destination register values are modified as a result of an arithmetic exception. The H.IP0F Register Fixup table (Table 2-4) shows how the different instruction types are modified. This capability exists for both base and nonbase mode tasks.

The arithmetic exception trap is enabled by setting the arithmetic exception bit (bit 7) of the task's PSD. By default, this bit is set when a task is activated. Instructions are provided in the base and nonbase mode instruction sets to manipulate this bit (see the EAE and DAE instructions). When the trap is disabled, only the condition code results are available to indicate that the exception has occurred, and the nature of the exception type.

Base mode provides the capability of further arithmetic exception handling within a task. By establishing an exception handler address within the task, the user provides the operating system an entry point to the task upon occurrence of an arithmetic exception.

Table 2-3
Task Interrupt Operation/Services Summary

	Task Interrupt Priority	Level 5		Level 4		Level 3		Level 2
	Task Interrupt Functions	Run Requests	Abort Requests	Message Requests	End-Action Run	End-Action Message	End-Action I/O	Break Requests
	Issue Request	M.SRUNR	M.BORT ABORT	M.SMSGR SEND	MPX	MPX	MPX	Hardware Break BREAK M.INIT
Sending Task Functions	Send Block	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Wait for Completion (wait)	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Establish End-Action Receiver	PSB	N/A	PSB	N/A	N/A	N/A	N/A
	Wait for Completion (no-wait)	M.ANYW	N/A	M.ANYW	N/A	N/A	N/A	N/A
	Call-Back Information	PSB	N/A	PSB	N/A	N/A	N/A	N/A
Receiving Task Functions	Establish Receiver	N/A	M.SUAR	M.RCVR	PSB	PSB	FCB	M.BRK
	Get Parameters	M.GRUNP	N/A	M.GMSGP	PSB	PSB	R1 points to FCB	N/A
	Receive Block	PRB	N/A	PRB	PSB	PSB	FCB	TY's UDT or Contents of T.BREAK
	Exit Receiver	M.EXIT M.XRUNR	M.EXIT	M.XMSGR	M.XREA	M.XMEA	SVC 1,X'2C'	M.BRKXIT
	Exit Block	RXB	N/A	RXB	N/A	N/A	N/A	N/A
	Wait for Next Request	RXB (if M.XRUNR)	N/A	M.SUSP M.ANYW	N/A	N/A	N/A	M.ANYW
	Disable Interrupt Level	N/A	N/A	M.DSMI	N/A	N/A	N/A	N/A
	Enable Interrupt Level	N/A	N/A	M.ENMI	N/A	N/A	N/A	N/A

Table 2-4
H.IP0F Register Fixup

Instruction Type	Exception Type	Destination Register Results
Floating Point Arithmetic includes: ADRFW ADRFD SURFW SURFD DVRFW DVRFD MPRFW MPRFD ADFW ADFD SUFW SUFD DIVFW DIVFD MPFW MPFD	Exponent underflow - positive or negative fraction	0
	Exponent overflow - positive fraction	Largest positive number (7F...F)
	Exponent overflow - negative fraction	Largest negative number (80...1)
	Division by zero	Largest positive number (7F...F)
Fixed Point Arithmetic includes: ADMB ADMH ADMW ADMD ADR ADRM ARMW ARMD ADI SUMB SUMH SUMW SUMD SUR SURM SUI DVMB DVMH DVMW DVR DVI RND	Any	No change (See specific CPU Reference manual)
Other includes: ABM ABR SLA FIXW FIXD SLAD LNW LND TRN TRNM	Any	No change (See specific CPU Reference manual)

The occurrence of an arithmetic exception with traps enabled causes the following events to occur:

1. The CPU generates a trap and transfers control to the arithmetic exception trap handler (H.IP0F).
2. The trap handler sets the T.EXCP flag in the TSA and determines what type of instruction caused the exception. If the exception was caused by one of the floating point arithmetic instructions, then the trap handler modifies the destination register values as described in Table 2-4.

3. If the exception occurred during a base mode task which has an exception handler established, and was caused by one of the floating point arithmetic instructions that causes register results to be changed, an argument list is constructed, and control is passed to the handler within the task via the CALL* instruction. When the task's exception handling routine is complete, control may be transferred back to the trap handler by using the RETURN* instruction.

   * This refers to Call/Return and argument passing standards established for FORTRAN 77/X32.

4. The exception trap handler restores all original register and condition code values, as well as the value of the task's current PSD, and allows the CPU to transfer control back to the task which caused the exception. Task execution resumes at the instruction following the trapped instruction.

Condition code values generated as a result of an arithmetic exception are defined in each of the CONCEPT/32 reference manuals.

2.7.9.1 Establishing Exception Handler

The M_SETEXA (Set Exception Handler) system service establishes an exception handler for base mode tasks. This service accepts as input either the address of the new handler to be established or 0 if the handler is to be ignored by the system. It provides as output the previous handler address. This allows a procedure to establish a handler and reset the previous handler when it no longer needs to handle exception conditions.

2.7.9.2 Changing a Return Address from an Exception Handler

The M_SETERA (Set Exception Return Address) system service can be called from an established exception handler to change the return address. This service accepts either the destination address where control is transferred upon exit from the handler or zero if the destination address remains unchanged (for example, execution is continued from the point of the trap).

2.7.9.3 Exception Handler Input Arguments

When an arithmetic exception handler exists within a task, it is entered from the exception trap handler via the CALL* instruction. The use of this instruction implies an argument list address in base mode R3. This list is a FORTRAN standard list containing the following arguments:

- value contained in the program counter (PC) at the exception
- arithmetic exceptions data array
- exception type and status indicator

The arguments are passed according to the FORTRAN standard for argument list construction. Assembly language programmers should take care in extracting desired information.

The program counter (PC) points to the instruction causing the exception. This may be a left or right halfword, or a fullword instruction. The C-bits in the program status doubleword (PSD) must be interpreted to determine the type of instruction.

The array contains information relevant to the arithmetic exception, for example, register contents at the time of the exception, the exception PSD, the register number to which the arithmetic modification was applied by the system arithmetic exception trap handler, and the condition codes at the time of exception.

The status value contains information used by the FORTRAN run-time routines. It includes the severity, system group, functional group, and type of exception.

The data structure of the argument list passed to the arithmetic exception handler of a base mode task has the following format:

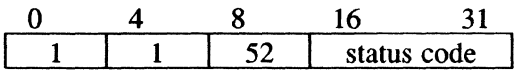
	0 7	8 15	16 23	24 31
Word 0	4			
1	Descriptor pointer			
2	Address pointer			
3	Array pointer			
4	Status pointer			
5	Address descriptor pointer			
6	Array descriptor pointer			
7	Status descriptor pointer			
8	3		0	
9	3		8	
10	1	2	0	
11	4			
12	8			
13	3		0	
14	Exception address			
15	20			
16-23	Exception general purpose registers			
24	Exception PSD 1			
25	Exception PSD 2			
26-33	Exception base mode registers			
34	Fixed register number			
35	Condition codes			
36	Status value			

Word	Description
0	number of words of pointer information that follow. In this example, the first word is a pointer to the descriptor list pointers for each argument in the list and the remaining three words are pointers to the arguments.
1	address of the descriptor vector. That contains one entry for each argument in the list. In this case, there are three. Each entry points to the information which describes the data type and size of each argument.
2	address of the word which contains the address of the instruction causing the arithmetic exception (word 14). Care must be taken if this parameter is used as the instruction may be a halfword or a fullword instruction.
3	address of an array of information collected when the exception occurred (words 15-35). The first word of the array contains the number of words in the array. This is FORTRAN standard. The following 21 words contain the 8 general purpose registers, the PSD, the 8 base mode registers, the register which was modified by the trap handler, and the condition codes at the time of the trap.
4	address of a status word supplied by the handler (word 36).
5	contains a pointer to the argument descriptor (word 8) for the first argument value. The argument value is contained in word 14.
6	contains a pointer to the argument descriptor (words 9-12) for the second argument. The argument value is contained in words 15 through 35.
7	contains a pointer to the argument descriptor (word 13) for the third argument. The argument value is contained in word 36.
8	contains the FORTRAN data descriptor for the first argument. The '3' in the left halfword indicates this argument is a word length integer data item. The '0' in the right halfword indicates that there is no additional descriptive information about this data item.
9-12	four words containing descriptive information for the second argument, the exception array. The '3' in the first halfword indicates this argument is a word length integer data item. The '2' in the right halfword indicates 2 more pieces of descriptive information follow. The '1' in the first byte of word 10 indicates that data is the size in bytes of one element of the argument. The '2' in the second byte indicates that data is the size in bytes of the entire argument. The '4' in word 11 indicates each element of the array is 4 bytes in length. The '80' in word 12 is the total byte length of the array (20 elements, 4 bytes each).
13	contains the FORTRAN data descriptor for the third argument. The '3' in the left halfword indicates this argument is a word length integer data item. The '0' in the right halfword indicates that there is no additional descriptive information about this data item.

Word	Description
14	contains the address of the instruction causing the exception. It may be the address of a fullword instruction, a left halfword instruction, or a right halfword instruction. The C-bits of the exception PSD must be interpreted to determine the type of instruction. The PSD C-bits are interpreted as follows:

Bit 30	Bit 31	Definition
0	0	fullword instruction
0	1	right halfword instruction
1	0	left halfword instruction
1	1	invalid

15	first word of the exception array. FORTRAN uses this word to contain the number of entries in the array for subscript validation.
16-23	8 words containing the contents of the 8 general purpose registers at the time of the exception. The destination register of the instruction causing the exception has had the modified value inserted.
24-25	2 words containing the PSD at the time of the exception. (Points to either 2 or 4 bytes past the instruction which caused the exception. (See word 14).
26-33	8 words containing the contents of the 8 base mode registers at the time of the exception.
34	contains the register number which was modified as a result of the arithmetic exception processing.
35	contains the 4-bit condition code value, extracted and right justified, which was contained in the exception PSD.
36	contains status information generated by the arithmetic exception trap handler in the following format:



Bits	Description
0-3	severity — value 1 = warning
4-7	system group — value 1 = O/S Support Library
8-15	functional Group — value 52 = Arithmetic exception
16-31	status code, contains:

Status Code	Description
1	exponent underflow, positive fraction
2	exponent overflow, positive fraction
3	exponent underflow, negative fraction
4	exponent overflow, negative fraction
5	divide by zero
6	fixed point exception if DQE.AF is set

2.7.9.4 Special Arithmetic Exception Processing and Ada Tasks

Any privileged task can set the DQE.AF bit to receive a status code 6 in word 36 of the argument list for fixed point exceptions. However, Ada tasks always have this bit set. Ada tasks receive arithmetic exception reporting for instructions that are usually not reported. These are: ABR, SLA, SLAD, TRN and TRNM.

The status codes in the right halfword of word 36 in the argument list are intercepted by the ADA module. These codes are reassigned as follows:

Status Code	Description	Reassignment
1	exponent underflow, positive fraction	2
2	exponent overflow, positive fraction	1
3	exponent underflow, negative fraction	2
4	exponent overflow, negative fraction	1
5	divide by zero	4
6	fixed point exception	0

2.7.9.5 Exception Handler Restrictions

Exception handlers execute with the following restrictions:

- Arithmetic exceptions encountered during execution of the user's arithmetic exception handler are processed by the system arithmetic exception trap handler, but do not cause the user's handler to be re-entered. Using the modified registers, execution is continued from the point of the trap or at the address specified to the set return address service.
- A user's arithmetic exception handler can be established only by base mode tasks.

2.7.9.6 Related Arithmetic Exception Information

The M.TSTE (Arithmetic Exception Inquiry) system service accesses the arithmetic exception flag, T.EXCP, in the T.BIT1 field of the task's TSA. The results of this service indicate whether an exception has occurred, and reset the T.EXCP flag. The output of the service is condition code results. Once the T.EXCP flag is set, it is not reset until this service is used or the task terminates. This is a nonbase mode system service.

2.8 CPU Dispatch Queue Area

The CPU dispatch queue is a variable length table built at SYSGEN and contains a maximum of 255 Dispatch Queue Entries (DQE's). Free DQE entries are linked into the C.FREE head cell in the standard linked list format. When a task is activated, a DQE is obtained from the free list and is used to contain all of the memory-resident information necessary to describe the task to the system.

For example, the task sequence number, owner name, load module name, TSA address, priority, and current state chain pointers are kept in the DQE, as are abort codes, message and run receiver queue addresses, etc.

Additional (swappable) information is maintained in the TSA. While a task is active, its DQE is linked to one of the various ready-to-run or wait state chains provided by the CPU scheduler to describe the task's current status. When a task exits, its DQE is again linked to the free list.

2.9 I/O Scheduling

I/O scheduling provides efficient service to I/O bound tasks while keeping the CPU busy with compute-bound tasks. This allows the fullest possible utilization of both the CPU and I/O devices.

A task that has been waiting for I/O to complete (SWTI or SWIO) is changed to an executable state at a priority slightly higher than a similar compute-bound task when the I/O completes, as described in the Situational Priority Increments section of this chapter. At that time, the CPU scheduler interrupts the execution of the compute-bound task so that the I/O-bound task can execute. The I/O-bound task requires minimal CPU time before initiating another I/O request and returning to the SWTI or SWIO state. The compute-bound task then resumes execution. The CPU scheduler automatically adapts to tasks that alternate between bursts of computing and bursts of I/O.

2.10 Swap Scheduling

The swap scheduler task (J.SWAPR) processes entries in the memory request queue (MRQ). It provides memory allocation and swap scheduling as appropriate to service individual requests for memory.

Refer to Reference Manual Volume I, Chapter 3 for swapping techniques used for demand page processing.

2.10.1 Structure

The swap scheduler is a memory resident, privileged task that is not operating system resident. It executes at the priority of the highest priority task in the memory request queue. The swap scheduler always occupies the first DQE. If a task requires memory, the swap scheduler maps the task's TSA on top of its address space.

The swap scheduler remains suspended until resumed by the executive in response to a swap scheduler event.

2.10.2 Entry Conditions

The swap scheduler task is normally suspended. It is relinked to the ready-to-run queue by the executive in response to a system service calling the executive to report a swap scheduler event. There are four basic types of swap scheduler events.

2.10.2.1 Dynamic Expansion of Address Space (M.GE/M.GD, M.MEMB)

When there is insufficient memory to satisfy a dynamic memory request for a task, the task is linked into the memory request queue and the swap scheduler is resumed.

Memory is allocated in 2KW increments on a CONCEPT/32. These increments are called map blocks.

2.10.2.2 Deallocation of Memory (M.FE/M.FD, M.MEMFRE)

When a task deallocates some or all of its memory, and the memory request queue is not empty, the swap scheduler is resumed. Those tasks in the MRQ are then allocated some or all of the deallocated memory.

2.10.2.3 Request for Inswap

When a currently outswapped task becomes eligible for execution, it is linked into the memory request queue. The swap scheduler is resumed to process the inswap request.

2.10.2.4 Change in Task Status

When a task which had been previously ineligible for swapping becomes eligible, the swap scheduler is resumed. Such status changes include the completion of an unbuffered I/O operation, the release of a lock-in-memory flag, or the expiration of a stage one time quantum.

2.10.3 Exit Conditions

The swap scheduler signals the executive when it cannot process any more outstanding requests, or when the memory request queue is empty. The swap scheduler is unlinked from the ready-to-run queue and placed in a special wait-for-memory-event state.

2.10.4 Selection of Inswap and Outswap Candidates

The swap scheduler attempts to allocate the memory required for the highest priority task in the memory request queue. If there is insufficient free memory, the swap scheduler examines the state queues on a priority basis, searching for the memory class and number of map blocks required.

When the first outswap candidate that satisfies any current memory request is determined, the task is outswapped.

When sufficient memory is available, the inswap process is initiated. The swap scheduler processes entries in the memory request queue until the queue is empty or until an available outswap candidate for a task requesting memory cannot be found.

Both outswap and inswap are serial processes which go to completion before the memory request queue is reexamined. Dynamic memory requests are similar to inswap requests, except that there is no associated disk file to read. Some tasks in the memory request queue can be queued for both inswap and a dynamic request. There must be sufficient memory for the inswap and dynamic requests before the inswap process can proceed.

2.10.4.1 Outswap Process

The outswap process is initiated when inswap or dynamic memory is requested. The outswap priority order can be specified by the system administrator. See the System Administrator chapter in MPX-32 Reference Manual, Volume III. The following table illustrates the default outswap priority order beginning with HOLD and ending with SQRT.

Default Outswap Priority Order	
Wait Queues	Ready-to-Run Queues
HOLD	SQ64
SUSP	SQ63
RUNW	SQ62
SWDV	SQ61
SWDC	SQ60
SWSR	SQ59
SWSM	SQ58
SWLO	SQ57
SWFI	SQ56
MRQ	SQ55
ANYW	RIPU
SWGQ	SQRT
SWTI	
SWIO	
SWMP	

The TSA of the outswap candidate is mapped into the swap scheduler and is used to construct a new address space which represents the swappable map blocks in a logically contiguous format. Then, the swap space is allocated and opened by the swap scheduler. For F-class swap devices, a single write request is given to IOCS. Command and data chains are built in the handler to perform the specified transfer.

Once output is complete, the memory is deallocated, and the memory request queue is re-examined to find the highest priority candidate for inswap.

2.10.4.2 Inswap Process

When sufficient memory is available, the swap scheduler allocates the memory required by the highest priority task in the memory request queue. If the request is simply a dynamic one, the swap scheduler adjusts the TSA of the requestor to reflect the newly allocated memory, and informs the CPU scheduler.

If the request requires an inswap, the swap scheduler reads the swapped image into the newly allocated memory. For F-class swap devices, a single read request is given to IOCS. Command and data chains are built in the handler to perform the specified transfer.

Once inswap is complete, the swap scheduler cleans its map and re-examines the memory request queue for the next inswap candidate.

2.11 Task Termination Sequencing

Three types of task termination are provided by the MPX-32 executive: exit, abort, and delete task execution.

2.11.1 Nonbase Mode Exit Task (M.EXIT)

The exit task service is called by a task that needs to terminate its execution in a normal fashion. The sequence of system processing on task exit is described in Table 2-5.

2.11.2 Abort Task (M.BORT)

The nonbase mode abort task service is called by a task that wants to terminate its execution in an abnormal fashion. It may also be initiated by the system when a task encounters a system trap condition (e.g., undefined instruction, privilege violation, or nonpresent memory) or by a system service because of a parameter validation error.

This service may also be asynchronously initiated by another task of the same owner name or by the OPCOM ABORT directive. The sequence of system processing on task abort is described in Table 2-5.

2.11.3 Delete Task (M.DELTSK)

The delete task service is called by the system for a task that encounters a second abort condition when processing an initial abort request. This service may also be initiated asynchronously by another task of the same owner name or by the OPCOM KILL directive. The sequence of system processing on task delete is reflected in Table 2-5.

2.11.4 Base Mode Exit Task (M_EXIT)

The base mode task entry structure allows base mode tasks to exit in a uniform manner. Exit sequences require a 0 or an ASCII status code to be placed in R0. Any entry into a subroutine may be exited by the execution of a RETURN instruction.

A base mode assembler task must exit any end-action receiver by a RETURN instruction. If the exit from end action requires a receiver exit block (RXB), the RXB address must be in R1.

Table 2-5
Task Termination Sequencing (EXIT, ABORT, and DELETE)

Task Has	System Action		
	Task Exit	Task Abort	Task Delete
Outstanding I/O	Defers processing until any outstanding I/O is complete.	Same as exit, except inhibits execution of user no-wait I/O end-action routines. Task abort is reflected in appropriate FCB(s).	Terminates all outstanding I/O.
Outstanding Messages in Receiver Queue	Unlinks all outstanding messages. Posts complete with abnormal status.	Same as exit.	Same as exit.
Outstanding No-wait Run Requests with Call Back	Defers processing until the destination task completes. The exiting task is placed in the ANYW state until the destination task has completed.	Defers abort processing until all requests are complete. Task abort status is reflected in run request parameter block.	Call backs are ignored.
Run Requests in Receiver Queue	Terminates the current run request and posts appropriate status in run request parameter block. Then activates a new copy of the task for next run request in queue, if any.	Same as exit.	Same as exit.

Continued on next page

Table 2-5
Task Termination Sequencing (EXIT, ABORT, and DELETE) *(Continued)*

Task Has	System Action		
	Task Exit	Task Abort	Task Delete
Task Abort Receiver	Not processed.	Transfers control to task after other steps taken above. Files are not closed. Devices and memory are not deallocated. (Remaining abort processing by system is discontinued.)	Not processed.
Files Open	Closes all open files automatically. Preserves integrity of both user and system files.	Same as exit.	Does not close files automatically. Preserves integrity of system critical files. User files are left as is.
Devices/ Memory Allocated	Deallocated automatically.	Same as exit.	Same as exit.

2.12 Task-Synchronized Access to Common Resources

MPX-32 provides the structure for tasks to voluntarily synchronize access to a common resource such as a disk file, a shareable device, a common data area, a shared/included procedure area, or any other physical resource.

The capability provided by MPX-32 is a general resourcemark mechanism. Each task using a marked resource must:

- use the M.RSML and M.RSMU (Resourcemark Lock/Unlock) services to synchronize access to a resourcemark with other tasks
- make the association of a particular resourcemark with an actual resource

MPX-32 provides: a table of resourcemarks that are currently in use, a mechanism for queuing tasks for each mark, and automatic unlock on a resourcemark when a task terminates (aborts, exits, or is deleted), if the task has not unlocked the resourcemark on its own.

A resourcemark is a decimal value from 1 to 64. Values 1-32 are for internal use, values 33-64 are available for customer use. The default size of 0 can be increased by using the SYSGEN RMTSIZE directive. When privileged tasks use the Lock/Unlock services, MPX-32 checks that the index value provided is within the range from 1 to the configured size of the resourcemark table. The system does not associate a particular resource with a particular resourcemark. Thus, if several tasks use synchronization service calls to gain access to a resourcemark and another task does not, the outside task gains the resource just as if no restrictions were active for it.

Tasks synchronizing use of resources are responsible for using resourcemarks that uniquely identify resources across the system. MPX-32 ensures only that a specified mark is within the legal numeric range.

To use resource marking, each cooperating task:

- uses M.RSML to lock the resourcemark
- performs the access which requires synchronization
- uses M.RSMU to unlock the resourcemark and release the highest priority task queued for the resourcemark

The task has several options available if the resourcemark is locked when it issues the M.RSML call. As specified in the call, it can:

- obtain an immediate denial return and go on
- wait until it can gain ownership of the lock
- wait until it can gain ownership or until a specified number of timer units have expired, whichever occurs first

If a single task uses more than one resourcemark, and it is synchronizing access to more than one resource, the user must exercise care to avoid deadlock situations; for example, task A is in wait for a lock owned by task B while task B is in turn waiting for a lock owned by task A.

A task using more than one resourcemark can avoid deadlocks by unlocking all locked resourcemarks if it cannot succeed in locking any one of them. The task then waits for the critical unlock to occur before reattempting locks on all the other resourcemarks in the set.

Sample Resourcemark Use by a Task

```
        PROGRAM T4
        M.REQS
        LIST NGLIST
T4      EQU             $
        M.RSML 33,0             LOCK RSM,INDEF.WAIT,NORM SWAP
        M.WRIT ABC              WRITE TO CRITICAL FILE
        M.RSMU 33               UNLOCK RSM
        M.EXIT
ABC     DATAW G'ABC'            LFC SETUP
        GEN 12/B 80,20/B(SBUF)
        REZ 6W
SBUF    RES 80B
        END             T4
```

2.13 MPX-32 Faults/Traps and Miscellaneous Interrupts

MPX-32 provides interrupt and trap processors for all standard interrupts and traps. A list of these interrupts with associated information is shown in Table 2-6.

Processing for trap levels 03, 04, 05, and 09 is dependent on the location of the instruction causing the trap. A system crash (M.KILL; not OPCOM KILL) results if the offending instruction is issued from a location within the MPX-32 system area. If the instruction is issued from a location within a task area, the task is aborted.

When a system crash occurs as a result of a trap handler entry, the CPU halts with the registers containing the following information:

Register	Contents
0	PSD word 0 (when trap generated)
1	PSD word 1 (when trap generated)
2	real address of instruction causing trap
3	instruction causing trap
4	CPU status word (from trap handler)
5	Crash code: MP01=X'4D503031' (Memory Parity Error - H.IP02) NM01=X'4E4D3031' (Nonpresent Memory - H.IP03) UI01=X'55493031' (Undefined Instruction - H.IP04) PV01=X'50563031' (Privilege Violation - H.IP05) MC01=X'4D433031' (Machine Check - H.IP07) SC01=X'53433031' (System Check - H.IP08) MF01=X'4D463031' (Map Fault - H.IP09) CP01=X'42543031' (Cache Parity - H.IP10) 32/67 and 32/87 AD01=X'41443031' (Address Specification - H.IP0C) HT01=X'48543031' (Privilege Halt Trap - H.IPHT)
6	Real address of register save block
7	C'TRAP'=X'54524150'

Table 2-6
MPX-32 Faults/Traps and Miscellaneous Interrupts

Relative Priority	Logical Priority	Dedicated TVL CPU	Dedicated TVL IPU	Description	System Action: PSD in OS area/ PSD in Task Area	Abort Code
00	00	80	20	Power fail trap	Halt/halt	N/A
01	01	84	24	Power on trap	Halt/halt	AU01
02/12	12	88	28	Memory parity trap	M.KILL/Abort	MP01
03/24	24	8C	2C	Nonpresent memory trap	M.KILL/Abort task	NM01
04/25	25	90	30	Undefined instruction trap	M.KILL/Abort task	UI01
05/26	26	94	34	Privileged violation trap	M.KILL/Abort task	PV01
06		98	38	SVC trap	Process SVC/ process SVC	See Note
07		9C	3C	Machine check trap	M.KILL/M.KILL	MC01
08		A0	40	System check trap	M.KILL/M.KILL	SC01
09		A4	44	Map fault trap	M.KILL/Abort task	MF01
0A			48	Undefined IPU instruction trap	M.KILL/Abort task	UI01
0C	0E	B0	50	Address specification trap	M.KILL/Abort task	AD02
0D	0D	B4	54	Console attention trap	Process Int./ process int.	N/A
0E	27	B8	58	CPU halt trap	M.KILL/Abort task	HT01
0F/29	29	BC	5C	Arithmetic exception trap	Not enabled/ record in TSA	N/A
10		C0	60	Cache memory parity error trap	M.KILL/Abort task	CP01
18	18	160	N/A	Real-time clock interrupt	Process Int./ process int.	N/A

Note:

SVC Abort Codes

SV01	Unprivileged task using M.CALL
SV02	Invalid SVC number
SV03	Unprivileged task using privileged service
SV04	Invalid SVC type
SV05	Unprivileged task using M.RTRN
SV07	Invalid SVC for base register operation

2.14 Real-Time Task Accounting On/Off

Disabling real-time task accounting allows users to improve context switch time for real-time tasks using any of the MPX-32 schedulers (EXEC, EXEC2, or EXEC3). The default mode is real-time task accounting on.

The overhead of multiple CD (command device) instructions that are executed every context switch can be avoided when real-time task accounting is turned off. When real-time task accounting is off and an IPU is present, then real-time task accounting is turned off in both processors.

Three levels of control are provided:

System Wide Default
: established at SYSGEN by the presence or absence of the MODE ONRA and MODE OFRA directives in the /PARAMETERS subsection of the //SOFTWARE section.

  The MODE ONRA is the default mode, which enables real-time task accounting. The MODE OFRA directive disables real time accounting. These directives do not affect tasks that are not real-time. These tasks always have accounting enabled.

OPCOM Override
: occurs with the !MODE ONRA and the !MODE OFRA commands.

  These commands provide a convenient way to override the SYSGEN defaults without performing a SYSGEN and RESTART to change the default accounting mode. The !MODE ONRA command enables real-time accounting. The !MODE OFRA command disables real-time accounting.

  This option has no effect on tasks that are not real-time. These tasks always have accounting enabled.

CATALOG Override
: occurs with the ENVIRONMENT ONRA and OFRA directives.

  The ENVIRONMENT ONRA directive enables real-time accounting regardless of the current default mode. The ENVIRONMENT OFRA directive turns real-time accounting off regardless of the current default mode. This option has no effect on tasks that are not real-time. These tasks always have accounting enabled.

3 Resource Management Overview

3.1 General Resource Management

A generalized resource management scheme means all resource operations work in a standard and predictable manner on every resource. A resource is any source of aid or support existing which is external to a task's body and is required by the task for that task to perform its function.

3.2 Support for Resource Types

The operating system recognizes two types of resources: physical and logical. A physical resource is any physical hardware supported by the operating system. A logical resource is any entity existing only because of a mechanism provided by software. Most often, the mechanism is merely a named and predictable data structure imposed on a physical medium.

3.2.1 Physical Resources

The primary physical resources supported by the operating system are: the central processing unit (CPU), computer memory (main storage), and I/O devices. In the support of physical resources, all resource functions are supported in the same manner. Definition and deletion of physical resources are accomplished by the system generation (SYSGEN) process. Resource attachment, access, and detachment are a subset of the functions allowed for logical resources. Physical resource inquiry is more primitive and resource dependent than logical resource inquiries. The attributes assigned to the management of physical resources are more resource dependent than the attributes associated with logical resources. Attributes of physical resources can only be modified by the SYSGEN process.

3.2.2 Logical Resources

The primary logical resources supported by the operating system are: disk volumes, directories, files, and memory partitions. In the support of logical resources, all resource functions are supported in the same manner. Definition and deletion of logical resources are accomplished by utilities or system services. Resource attachment, access, detachment, inquiry, and attribute modifications are provided for logical resources and are implemented by system services.

3.3 Support for Resource Functions

To support all resources in a similar manner, all functions required to manage resources must be provided and those resources must operate in a similar manner. The following is a list of the functions provided:

Function	Description
Resource creation	defines a resource to the operating system
Resource deletion	removes the definition of a resource from the operating system
Resource attachment	connects to a resource for the purpose of using the resource
Resource access	uses a resource or transfers data to or from a resource
Resource detachment	disconnects a resource so it can be used by others
Resource inquiry	inquires about a resource to determine specific information about it
Resource attribute modification	modifies the attributes of a resource to change its operational characteristics

Detailed descriptions of these functions are in the following sections.

3.3.1 Resource Creation

Before any resource can be used by a task in the operating system, the required resource must be defined to the operating system. Utilities are provided for defining resources. In most cases, the resources can be defined by directives or services issued to the operating system. When a resource is created, all resource attributes are defined.

3.3.2 Resource Deletion

The resource definition must be deleted so the resource can no longer be used. This is accomplished by the use of a utility program. Usually, resources can be removed using directives or services provided by the operating system.

3.3.3 Resource Attachment

Resource attachment is the process of securing a resource for use by a task. Commands and/or service requests are issued to the operating system to attach a resource. When a resource is to be attached, various parameters are specified indicating how the resource is to be used. The directive or service to attach a resource is unique to each type of resource. For example, volumes are mounted, memory partitions are included, and files/devices are assigned.

There are two types of resource attachment provided by the operating system, static and dynamic.

3.3.3.1 Static Allocation

Static allocation is invoked by declaring a task's resource requirements when the task is cataloged or activated. Static allocation serves several purposes. Most importantly, this form of allocation guarantees that a cataloged load module, when activated, has all the resources required before it begins execution. Secondly, by declaring all of a task's resources when it is cataloged or activated, the operating system can match sets of resource requirements among all tasks in the activation state and more effectively manage resources. Static allocation enables the operating system to allocate sets of resources. This avoids deadlocks that can occur when a task requires multiple resources and must dynamically allocate each one. Statically allocated resources can be overridden at task activation.

3.3.3.2 Dynamic Allocation

In some applications, a task does not know what resources it requires until execution begins. For this reason, dynamic resource allocation is provided. Dynamic allocation is invoked by service requests from an executing task.

3.3.4 Resource Access

Resource access is the process of transferring data to or from a resource, positioning a resource, or otherwise manipulating a resource. Various applications require many levels of access to a resource. The operating system provides the following levels of resource access. The levels are described in order from the most device-dependent to the least device-dependent access levels.

3.3.4.1 Device Level

The operating system provides integration of device-dependent I/O drivers. The user is not required to design and code an MPX-32 I/O device handler. However, a user-supplied I/O driver can be integrated into the operating system by the SYSGEN utility.

3.3.4.2 Execute Channel Program Level

To perform device-dependent I/O operations where the operating system queues the I/O requests, starts the I/O requests, and processes the operating system termination functions, the user can build and execute physical or logical channel programs. These channel programs can only be executed on devices that use extended I/O (XIO) protocol.

3.3.4.3 Logical Device Level

With this level of access, the user can transact with a device while using specific physical capabilities offered by the device. Logical device I/O supports applications which require the use of a specific device for a capability provided explicitly by the device.

With this level of access, a user performs I/O requests using a file control block (FCB). The data format inhibit option must be set in the FCB to gain access to the device at this level.

3.3.4.4 Logical File Level

With this level of access, a user achieves a degree of device independence. When device access is performed at this level, device-dependent characteristics are masked from the user. This access level supports applications which require the illusion of device commonality and independence.

3.3.4.5 Blocked Level

This is the highest level of device access provided by the operating system. Explicitly intended for use by utility programs requiring the highest degree of device commonality or sameness, blocked I/O works only with magnetic tape and disk media. For all operations to perform identically regardless of which type of media is manipulated, this access level emulates the operation of magnetic tape on both types of media. All operations that are valid for magnetic tape media are provided. The user can issue rewind, write end-of-file, advance file, backspace file, advance record, backspace record, read record, and write record operations.

With blocked I/O, the user must access the media in a sequential manner. Records can be transacted with the media in lengths of 254 bytes or less; longer records are truncated. The operating system automatically performs intermediate record blocking and buffering to or from the media.

For the maximum device independence, use the following subset of allowable operations: rewind, read record, and write record.

Note: Append access is available on disk media but is not allowed on magnetic tape media.

3.3.5 Resource Detachment

Resource detachment allows attached resources to be released and made available for use by other tasks. When resources are detached, other tasks that can be queued awaiting the availability of the resource are resumed to contend for attachment to the resource.

Resources are detached explicitly by the appropriate dismount, exclude, or deassign functions. Additionally, any resources a task has attached at the normal or abnormal termination of the task are automatically detached by the system.

Resource detachment may cause the system to perform clean-up operations, such as purging partially-filled blocking buffers and releasing exclusive locks outstanding on the resource.

3.3.6 Resource Inquiry

Resource inquiry is provided so tasks can determine the attributes of a resource.

3.3.6.1 Inquiry of Unattached Resources

At times it is necessary for a task to inquire about a resource. Given the name or some other legal identifier, directives and services are provided to return information about resources. The operating system provides this information through a resource descriptor (RD). At a place common to all resource descriptors, the inquirer can determine the type of resource. Once this information is obtained, the inquirer can examine resource type-dependent data in the descriptor for more specific information. Usually, such inquiries are made before the resource is attached.

3.3.6.2 Inquiry of Attached Resources

This type of inquiry is most often used when a resource has been statically attached. It determines the logical or physical attributes of the connection, such as the access modes, access level, physical device address, and parameters. The user must furnish the logical file code, file control block (FCB) address, or allocation index to identify the desired resource.

3.3.7 Resource Attribute Modification

At times, it is necessary to modify the protection and other access attributes of a resource. Resource attribute modification, like resource inquiry, deals with operating system data structures. The user of these functions should be familiar with the format of these data structures. Also, it is recommended that user-supplied subroutines act as a common interface to the functions. In this way, the user is less sensitive to changes in system structures.

3.4 Resource Attributes

All logical resources have attributes. The attributes of resources control how the resources are managed and determine who can use them.

The operating system ensures that all logical resources are defined in directories; for example, by providing names. Protection is applied to resources to determine who may use them and how they can use them. Resources can be shared (used by more than one task at the same time). Resources can otherwise be declared as nonshared (used by only one task at a time). Another set of attributes determines how the resource can be accessed; for example, data can be read from but not written to the resource, only certain areas of the resource can be written to, the resource can be deleted, etc.

3.4.1 Protection

The protection provided by the operating system for logical resources is organized so that a resource can be managed by the following:

Class	Description
Owner	is the person creating a resource. When the resource is created, the owner establishes all attributes for the resource. The owner can specify which project group and others can access the resource and what their access capabilities are.
Project Group	is the name of a group of users allowed to access the resource.
Others	are users of a resource who are not the owner or members of the project group.

A resource can be defined and managed as applicable to each class.

Protection is supplied for environments where desired. Since protection can be harmful when it is not administered properly, the user is advised to protect resources only to the level required. By default, owner and project group privileges are equal and all owners belong to the same project group. This default method of operation allows all users of the system to attach to all resources defined to the system.

When a task attempts to attach a resource, the system determines the owner, a member of a project group, or an other arbitrary user is making the attempt to attach a resource, the associated owner name is checked first, then the project group name. If the task does not match either of the first two checks, the task is given the access rights associated with an other arbitrary user. Otherwise, the task is given the access rights for the level that was matched.

3.4.2 Shareable Resources

A resource can be defined as shareable when it is created. Shareable resources can be attached to more than one task at the same time.

When a shareable resource is attached, the requesting task indicates how the resource is used. A shareable resource can be used in three modes: exclusive, explicit, and implicit. The resource can only be attached in one mode at any time.

3.4.2.1 Exclusive Use

When a task requires exclusive use of a shared resource, the task can request it when attaching to the resource. A resource attached for exclusive use can only be used by the task that was granted the exclusive attachment. Once a resource has been attached for exclusive use, other tasks requesting attachment can be denied or enqueued until the resource becomes available.

A task may require exclusive access to a resource it is already attached to and may be currently sharing with other tasks. In this case, the task must call the M.LOCK service. The M.LOCK service determines whether the caller is the only task attached to the resource. If so, the caller is immediately given exclusive access. Otherwise, the caller is denied or enqueued until it receives exclusive access.

3.4.2.2 Explicit Use

When multiple tasks require simultaneous attachment to the same shareable resource, the tasks can attach to the resource for explicit use. With explicit use, the attached tasks can use the resource in a way that can destroy recorded data. It is the responsibility of tasks using a resource in this mode to ensure that the integrity of the data recorded on the resource is preserved.

The following mechanisms allow explicit users to preserve data integrity:

- A task can gain exclusive access to a resource by calling the M.LOCK service.

  In this case, the M.LOCK service enqueues the caller until exclusive access can be granted.
- A set of tasks can synchronize on access to the resource by calling the M.SYNC and M.UNSYNC services.

  Synchronization locking does not guarantee exclusive access to a resource; it is merely a semaphore that is locked. The semaphore for the resource is returned when the resource is attached and must be used as an input parameter to the M.SYNC and M.UNSYNC services.

  Another type of semaphore, the resourcemark, can be used for synchronizing access to a resource or a set of resources. The use of semaphores requires that all tasks attached to a resource in explicit use mode cooperate to preserve the integrity of the resource.

3.4.2.3 Implicit Use

When a task attaches to a shared resource but does not explicitly declare exclusive or shared use, the resource is attached for implicit use. Implicit use is the default usage mode for all attachment to resources. In this mode, the operating system automatically allows the resource to be accessed by multiple tasks attached in compatible access modes. With compatible access modes, the following access combinations are allowed: multiple readers, multiple readers with a single writer, single writer only, and two simultaneous writers.

3.5 Resource Access Attributes

All resources have a set of attributes to determine how the resource can be accessed. This section defines the access attributes for each logical resource and defines the purpose of these attributes.

3.5.1 Access Attributes for Volumes

Access to resources on a volume is determined by attributes assigned when the volume is mounted and by the volume type. Refer to the Volume section in Chapter 4 of this manual.

3.5.2 Access Attributes for Directories

Directories have attributes determining how they are managed. These attributes can be specified for each user class (owner, project group, and others). The access attributes are defined when a directory is created. Access attributes for directories are:

Access	Description
Read	A directory with read access can be attached like a file with read-only access. A user assigning a directory in the read mode must be familiar with the format of directory entries to extract meaningful information. Read access lets the contents of a directory be presented by the LOG service or directive with wild card characters. If read access is not allowed, a directory cannot be logged.
Add entry	New directory entries can be added to a directory with add entry access.
Delete entry	Directory entries can be deleted from a directory with delete entry access.
Delete directory	A directory with delete directory access can be deleted when it does not contain any active directory entries. Utilities, directives, and services are provided to delete entries from directories.
Traverse	A directory can be searched when a pathname is being executed by the system if it has traverse access. For example, a resource can be located in a directory. The contents of a directory cannot be presented with the LOG service or directive if wild card characters are used and the user only has traverse access to the directory.

3.5.3 Access Attributes for Files

Files have attributes determining how they are managed. These attributes can be specified for each class of user: owner, project group, or other users. The access attributes are defined when a file is created. This applies for both temporary and permanent files. Access attributes for files are:

Access	Description
Read	A file with read access can be attached for read only access.
Write	A file with write access can be attached for read/write access. This mode establishes all new data contents for a new or existing file. In this mode, unused extensions to a file are automatically deleted when the file is closed.
Modify	A file with modify access can be attached for read/write access. This mode modifies the data contents of an existing file. When accessed in this mode, files cannot be automatically extended.
Update	A file with update access can be attached for read/write access. This mode modifies the data contents of an existing file and appends new data to the file. When accessed in this mode, files can automatically extended.
Append	A file with append access can be attached for read/write access. This mode appends new data contents to an existing file. When accessed in this mode, files can be automatically extended.
Delete	A file with delete access can be deleted if the directory containing the file's directory entry allows delete entry access.

3.5.4 Access Attributes for Memory Partitions

Memory partitions have attributes determining how they are managed. They can be specified for each class of user: owner, project group, or other. The access attributes are defined when a memory partition is created. Access attributes for memory partitions are:

Access	Description
Read	A memory partition with read access can be attached (included) for read-only access.
Write	A memory partition with write access can be attached (included) for read/write access.
Delete	A memory partition with delete access can be deleted if the directory containing the partition's directory entry allows delete entry access.

3.6 Management Attributes

All logical resources have a set of applicable management attributes regardless of which user class is attached. Management attributes for resources are described in this section.

3.6.1 Extension Attribute

Extension attributes apply only to permanent and temporary files.

3.6.1.1 Manual Extension Attribute

Manual extension attributes apply only to files. They enable a file to be manually extended by the extend service or directive.

3.6.1.2 Automatic Extension Attribute

Automatic extension attributes apply only to files. They enable a file to be extended automatically when data is written to the file. Automatic extension is subject to restrictions inherent to access modes. For example, a file attached in the modify mode cannot be extended.

3.6.2 Contiguity Attribute

Contiguity attributes apply only to extendible files. This attribute informs the operating system that the file should be contiguous. When a file has this attribute and is to be extended, the operating system attempts to allocate the new segment contiguous to the last segment. If the contiguous extension fails, the system attempts to allocate the new segment at any available location on the same volume. The Volume Manager RESTORE directive can be used to attempt to restore a discontiguous file contiguously if it contains the contiguous attribute.

3.6.3 Maximum and Minimum Extension Attributes

The maximum and minimum extension attributes can only be described by a detailed explanation of the extension algorithm. When a file is to be extended, the presence of a dynamically user-supplied value, a minimum increment, and the maximum increment is verified. The following chart shows the results of this verification.

Supplied Value	Minimum Increment	Maximum Increment	Result
No	No	No	file is not extended
No	No	Yes	maximum extension attempted
No	Yes	No	minimum extension attempted
No	Yes	Yes	maximum extension attempted first, minimum attempted second
Yes	No	No	supplied value attempted
Yes	No	Yes	supplied value attempted first, maximum extension attempted second
Yes	Yes	No	supplied value attempted first, minimum extension attempted second
Yes	Yes	Yes	supplied value attempted first, maximum extension attempted second, minimum extension attempted third

3.6.4 Maximum File Size Attribute

Maximum file size attributes apply only to files and can be specified when a file is created. If specified, a file does not become larger than the size specified. If a maximum file size is not specified, a file can be extended a maximum of 31 times or until file space cannot be acquired, whichever occurs first.

3.6.5 Shared Attribute

Shared attributes apply to files and directories and can be specified when the resource is created. If specified, the resource can be attached and accessed by more than one task. If shared is not specified, the resource can only be attached and accessed by one task at a time. Memory partitions are always given the shared attribute.

3.6.6 End-Of-File Management Attribute

End-of-file management is an attribute of both blocked and unblocked files. It controls how MPX-32 performs end-of-file accounting. The end-of-file management attribute is set at the time of file creation.

All files are created with EOFM=T/Y, (either T or Y may be specified) unless EOFM=F/N (F or N) is specified. For files created with EOFM=T/Y, end-of-file management is performed through file descriptor accounting. For files created with EOFM=F/N, EOF accounting is managed via an EOF indicator within the file contents.

3.6.7 Fast Access Attribute

Fast access applies only to files and can be specified when a file is created. This attribute enables files defined on a volume to be attached in one disk access through the file identifier (RID). The RID identifies the volume where the file resides, the block number of the file's resource descriptor (RD), and the creation date and time of the file. Files created with the fast access attribute always retain their original RID as long as they remain defined on the volume. A file explicitly deleted and subsequently recreated is assigned a new RID.

System services using a resource create block (RCB) can create files with the fast access attribute by setting the appropriate bit in the RCB. See the Resource Create Block (RCB) section of Chapter 5 for information on the RCB.

When a Volume Manager RESTORE or COPY directive is performed on a file created with the fast access attribute, the file retains its original RID. See MPX-32 Reference Manual Volume II, Chapter 3.

3.6.8 Zero Attribute

Zeroing applies only to files and can be specified when a file is created. File space is prezeroed when a file is created or extended with this attribute.

3.6.9 File Type Attribute

File type applies only to files. This is a 2-digit hexadecimal number that can arbitrarily classify resources. File type codes are:

Value	Description
00-39	available for customer use
40-5F	reserved for system
60-9F	available for customer use
A0-AF	reserved for system
B0	base mode object file
BA	base mode shared image (or BASIC file)
BB	base mode object library file
BC	base mode macro library file
BE	base mode load module file
C0	spooled output file
CA	cataloged load module
CE	MPX-32/COFF executable image
CF	MPX-32/COFF shared image
D0	memory disk save task (J.MDSAVE) file
DB	symbolic debugger command file
ED	saved text editor file
EE	stored text editor file
FD	translated help file
FE	text editor work file
FF	SYSGEN generated file

3.6.10 No-Save Attribute

No-save applies only to files and can be specified when a file is created. A file with this attribute cannot be saved by the Volume Manager SAVE directive unless the SAVN parameter is Y.

3.7 Operating System Memory Allocation

Unless extended execution space is specified, MPX-32 occupies the lower portions of each task's address space. This allows MPX-32 to run mapped with each task. On CONCEPT 32/2000 systems running a mapped out image, MPX-32 runs unmapped and each non-base task may choose to run with MPX-32 mapped into or mapped out of its address space.

MPX-32 maintains lists of available memory for allocation to tasks and for I/O that requires intermediate buffering.

3.7.1 I/O Buffer and I/O Queues

The system memory pool is an area of memory which is contiguous to the resident system and has a size specified at SYSGEN by the POOL directive. The entire memory pool is write-protected from the unprivileged task and is intended for use exclusively by system services. The memory pool is mapped into the address space of each task and is allocated in doublewords. The maximum size of any entry is 192 words. Typical system uses of the memory pool area are:

- I/O queues — Approximately 26 words are allocated when IOCS queues a request and deallocated when post-I/O processing is complete.
- Message or run-request buffers — Up to 192 words are allocated by the M.SMSGR or M.SRUNR services and deallocated when receiver processing is complete.

3.7.2 Blocking Buffers for Blocked I/O

File assignments for permanent files and devices optionally specify that a file is blocked or unblocked. The default is blocked. If blocked, blocking buffers for the files are allocated at load time in the task service area (TSA). The Catalog BUFFERS directive may be used to provide additional blocking buffer space for dynamically allocated, blocked files.

3.7.3 Large Buffers for Blocked Files

Large blocking buffers contain two or more 192-word blocks. The total number of buffers should not exceed 247.

When using the TSM $ASSIGN directive with the BBUF parameter, large blocking buffers can be allocated at load time in the task service area (TSA). The BBUF parameter is only valid with a TSM $ASSIGN directive.

User-supplied large blocking buffers in multiples of 192-word blocks can also be created using a 16-word expanded FCB. Byte 0 of word 15 contains the number of 192-word blocks within the large blocking buffer. The address of the buffer is placed in bytes 1 through 3. When byte 0 does not specify a number of blocks, one blocking buffer is automatically allocated.

3.8 Memory Classes

When a nonbase mode task is cataloged, the user can specify the class of memory required at run time. Memory classes E, H, and S are established at SYSGEN time, and there are no limitations for the positioning or sizes of these classes. Memory class D specifying SelBUS (DRAM) memory is configured above SRAM memory. The Cataloger ENVIRONMENT directive indicates the class of memory with the following parameters:

Parameter	Results
S	execution delayed until class S, H, or E is available (default)
H	execution delayed until class H or E is available
E	execution delayed until class E is available
D	SelBUS (DRAM) memory (CONCEPT 32/2000 only) This class of memory is non-executable and is not recognized by the Cataloger ENVIRONMENT directive.

When the requested memory class is not installed, the first available lower class is allocated to that task. If an excessive request is made, the requestor is aborted.

If there is no ENVIRONMENT parameter for memory class, tasks are loaded into any memory class available.

Base mode tasks are loaded into any memory class available.

3.9 Memory Allocation for Tasks

The unit of memory allocation is called a map block and is 2KW on a CONCEPT/32. All user tasks are discontiguously loaded into a whole number of physical map blocks, utilizing the mapping mechanism to create their contiguous logical address space. No partial map blocks are allocated.

This allows user tasks to dynamically expand and contract their address space by using the memory management service calls described in Chapter 6.

The unit of memory protection is called a protection granule and is 512 words. Thus, it is possible to protect a task's TSA even though it is in the same map block as the data section (DSECT or read/write).

On CONCEPT 32/2000 systems, memory allocation for tasks is handled differently when a mapped out image is resident and demand page processing is in effect. Refer to the next section for details.

3.9.1 Demand Page Processing (CONCEPT 32/2000 Only)

Demand page processing provides better physical memory management for a multitask environment and allows for execution of tasks which are larger than the available physical memory. When demand page processing is in effect for a task, it is loaded on demand into memory in map block increments. As additional logical address space is referenced in a task, the map block (page) needed to satisfy the address is brought into memory (paged in) and added to the task's working set of map blocks. The working set consists of the physical memory mapped into the logical space of the task.

As pages are no longer referenced by the task within a specified amount of time, they are considered aged and are removed from the working set. Aged pages are either modified or unmodified. Modified aged pages are linked to the task's page-out queue or shared memory page-out queue for writing to the swap volume. After they are paged out to the swap file, the physical memory can then be added to the free list (freed) for reuse. Unmodified aged pages are simply freed. Freed pages which still contain valid information from the last allocation can be retrieved. Paged out and freed pages are not part of the task's working set.

When a new memory address not currently in the working set is referenced (page fault), the page satisfying that address is sought, first from the free list or page-out queue, then from the swap file or load module. Retrieving a page from the free list or page-out queue improves efficiency by avoiding I/O from the load module or swap file. Data is not read from the disk because the page exists in memory as it did when queued for page-out.

Demand page is the default processing mode on a mapped out system image. For information on installing a mapped out system image, refer to Reference Manual, Volume III, Chapters 2 and 3. Demand page is not supported on mapped in images.

When demand page is supported, tasks are eligible for demand page processing when:

- the task is absolute (no relocation is necessary)
- the task's TSA is in extended memory and MPX-32 is mapped out of the task's address space

The SYSGEN DEMAND directive can be used to specify that only those tasks cataloged or linked as demand page are demand paged. Or, this directive can be used to change the default priority level range in which an eligible task is demand paged. Additional SYSGEN directives are available to fine tune your demand page processing or inhibit it. Refer to Reference Manual, Volume III, Chapter 7 for details.

3.9.2 Static Memory Allocation

The Cataloger determines the size in protection granules of a cataloged load module. The Cataloger ALLOCATE directive may be used to specify additional bytes of memory. The size of the TSA is determined at activation time and rounded up to a number of protection granules. This value is added to the cataloged requirement to determine task size. Additionally, the TSM $ALLOCATE directive may be used to specify the total task size of the DSECT. The final sum is rounded up to a map block increment. For information on static versus dynamic shared memory, see Table 3-1. Information on memory partition applications for nonbase tasks is in Table 3-2.

Table 3-1
Static versus Dynamic Shared Memory

Characteristics	Static Partitions	Dynamic Partitions	Shared Image
Logical addresses	Fixed at SYSGEN	Fixed by Volume Manager CREATE COMMON	Determined at link time
Physical addresses	Fixed at SYSGEN	Variable	Variable
Allocation unit	2K words	2K words	2K words
Time of allocation	SYSGEN	Run time by M.INCLUDE	Automatic by activation or M_INCLUDE
Time of deallocation	Never	When allocation count=0	When allocation count=0
Inclusion	Automatic by activation or M.INCLUDE	Run time by M.INCLUDE	Automatic by activation or M_INCLUDE
Exclusion	Automatic by exit or M.EXCLUDE	Automatic by exit or M.EXCLUDE	Automatic by exit or M_EXCLUDE
Owner names or task numbers	None	Established by M.INCLUDE caller	None
Swapping	Never Swapped	Swappable when user count=0	Swappable when user count=0

Table 3-2
Memory Partition Applications for Nonbase Mode Tasks

Characteristics	Global	Datapool	Extended Common	CSECTs
Cataloger resolves references	Yes	Yes	No	Yes
Compiler resolves references through extended bases	No	No	Yes	N/A
Must be logically below 128KW	Yes	No	No	Yes
Variables are order dependent	Yes	No	Yes	N/A
Static	Yes	Yes	Yes	No
Dynamic	Yes	Yes	Yes	Yes

3.9.3 Dynamic Address Space Expansion/Contraction (M.GE, M.FE, M.GD, M.FD, M.MEMB, M.MEMFRE)

A nonbase mode task can expand and contract its execution and extended data space through the system services. The M.GE service appends a map block to the user's execution space starting at the end of the DSECT. M.GE can be used more than once to obtain additional map blocks, as long as they are available in the task's logical address space. The M.FE service frees the most recently obtained map block; for example, it works in the opposite order of M.GE. The M.GD service appends a map block to the user's extended indexed data space, starting from 128KW. Like M.GE, it can be used more than once. The M.FD service frees the most recently obtained map block from the extended indexed data space; for example, it works in the opposite order of M.GD. The M.GE and M.GD services do not apply to base mode tasks.

The M.MEMB service provides dynamic memory allocation capabilities for the user. Memory is allocated in byte increments on doubleword boundaries. Allocation starts at the next doubleword boundary following the user's last loaded address (T.END). Any number of bytes can be specified up to the maximum available in the user's logical address space. Each call to the service provides contiguous allocation for the requested amount. Areas allocated by subsequent calls are not contiguous with previously allocated areas. Allocated areas may or may not be in extended memory. The user should operate in extended mode (SEA) when addressing these areas. The M.MEMFRE service frees memory, in random order, obtained from the M.MEMB service. These two services cannot be used with the M.GE, M.GD, M.FE, and M.FD services.

3.9.4 Extended Indexed Data Space for Nonbase Mode Tasks

MPX-32 provides limited support for logical addresses above 128KW for nonbase mode tasks. The following restrictions apply to the use of this address space:

- instructions cannot be executed in this logical space
- the user must reference this space by index registers. Negative offsets are invalid in the word address field of any instruction as long as the indexed addressing mode is active.
- no data initialization facilities are provided for this logical space
- the user must dynamically request this logical space to be mapped with the memory management system services
- global is not supported in extended data space

3.9.5 Intertask Shared Global Memory and Datapool Memory (M.INCLUDE, M.EXCLUDE)

Intertask shared memory is provided through Global and Datapool memory partitions and shared images. A task may include up to one hundred Global regions (GLOBAL00 - GLOBAL99) plus up to 101 Datapools (DATAPOOL, DPOOL00-DPOOL99).

Global and Datapool partitions can be defined using SYSGEN or the Volume Manager utility. Shared images are created by the LINKER/X32. See the Shared Images section of Chapter 4 for shared image information.

Partitions created at SYSGEN are considered permanently allocated and are assigned both physical and logical memory attributes applying to any task that references the partitions. This type of allocation is called static allocation. The static Global and Datapool partitions are defined in integral numbers of pages.

Both Global and Datapool partitions are located in an integral number of physical map blocks starting on a map block boundary. Areas are mapped into user space when required. See the Static Memory Allocation and Dynamic Address Space Expansion/Contraction sections of this chapter for a description of how MPX-32 maps areas into user space.

Write protection is available and prevents the user from storing into a common area that has write access. Dynamic shared memory partitions are defined with the Volume Manager.

There are several key distinctions between statically and dynamically allocated common partitions:

- Static partitions are fixed in physical memory even when no task is sharing them. Dynamic partitions are deallocated when their allocation count equals 0.
- Statically allocated partitions are invoked on a system-wide basis. Dynamically allocated partitions are based on a subsystem concept, where tasks issue an M.INCLUDE request. A particular common partition, such as Datapool, can be defined concurrently in several such subsystems. However, each subsystem has a physically unique partition.
- Dynamically allocated common partitions can be excluded from a task by the M.EXCLUDE system service. The user can elect to subsequently include another dynamically allocated common area by M.INCLUDE. Statically allocated partitions are not supported by the Volume Manager.
- All logical references to common, whether statically or dynamically allocated, are resolved by the cataloger. The logical address of a system common partition is fixed when the partition is defined.
- Load modules from one MPX-32 configuration are compatible with another even if Global or Datapool are allocated different physical addresses. The only compatibility requirement is that both systems employ the same logical conventions.

Figure 3-1 illustrates a relatively complex view of the relationship between logical address spaces and statically and dynamically allocated common partitions. The figure also introduces the allocation considerations for shared procedures, which are described in the section which follows.

In brief,

- Tasks A, B, and C reference a static Datapool partition.
- Tasks A and B use an M.INCLUDE service for dynamic GLOBAL10.
- Tasks A, B, and C use M.INCLUDE for GLOBAL02.
- Task D shares no memory or code with other tasks. Map blocks seven and up are available to the task.
- Tasks A and B are shared. They have CSECT mapped at the same location in each logical address space.

3.9.6 Shared Procedures for Nonbase Mode Tasks

MPX-32 supports shared procedures. A catalog parameter specifies a shared procedure that must consist of a CSECT (pure code and data) and a DSECT (any impure data). When a shared procedure is activated for the first time, the system loader reads both the shared procedure section and the impure data section into memory.

The shared procedure section is mapped like an M.INCLUDE service request. Every subsequent activation of the shared procedure causes only the impure data section of the shared procedure to be loaded since the procedure section is already in memory, has not been altered, and can be mapped like an M.INCLUDE service request.

Shared procedures, such as shared memory areas, have an allocation count to prevent premature deallocation of the memory they occupy and a user count to control the swapping of these partitions.

Shared procedure space is allocated in the highest available logical address. (See Figure 3-1.)

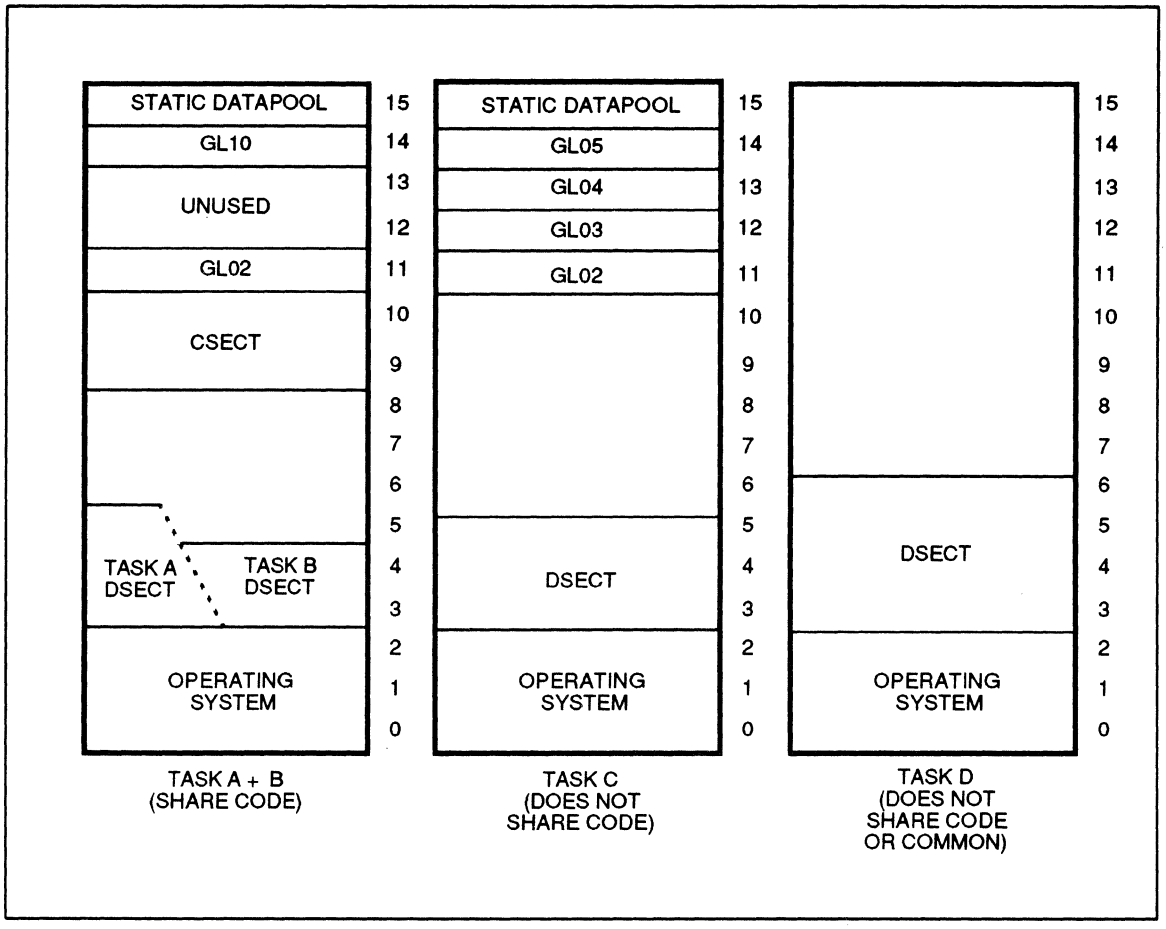


Figure 3-1
Sample Allocation of Common Memory Partitions and Common Code

3.9.7 Multiprocessor Shared Memory

87D12W07

Multiprocessor shared memory is memory that is shared between systems. This portion of memory must be managed by the user. If not properly SYSGENed, it is possible for MPX-32 nonresident tasks to be allocated memory in the shared memory sections. This can allow corruption of the nonresident tasks by the other systems.

It is recommended that multiprocessor shared memory be used as follows:

- For memory that is shared between systems, create a static memory partition in the multiprocessor shared memory via SYSGEN. The partition can be any size.
- Any remaining memory can be allocated to MPX-32 tasks. This memory should be SYSGENed exclusively for a particular system. This memory can be divided between systems as long as each area can be accessed by only one system. To accomplish this, SYSGEN each area as present in one system and non-present in all other systems.

3.10 Extended MPX-32 (Expanded Execution Space)

Extended MPX-32 is an optional mode of operation that allows a portion of the MPX-32 operating system to be positioned into a task's extended memory. Using extended MPX-32 results in more execution space for nonbase tasks.

Extended MPX-32 creates a split image that divides the operating system into two sections:

- The nonextended section of the split image is nonbase code that is mapped into the lower 128KW of user task space below the TSA.
- The extended section of the split image is translated into base code. This allows part of the operating system to be removed from the lower 128KW task space and placed in the extended address area where only data could previously be placed.

As a result of the split image, additional nonbase execution space is available in the lower 128KW of the task's logical address space.

Extended MPX-32 can be located:

- at a specific logical map block address
- at the end of logical extended memory (MAXADDR)
- between the task service area and the task DSECT (MINADDR)

Using extended MPX-32 at MAXADDR increases the user's task execution space by several map blocks.

The user can position extended MPX-32 by using the:

- SYSGEN EXTDMPX directive
- CATALOG EXTDMPX directive
- TSM EXTDMPX directive

The position of extended MPX-32 is determined during task activation, and is then fixed. Each task chooses the position of extended MPX-32.

Note: Extended MPX-32 can only be used on CONCEPT/32 systems that have base mode capability. Attempts to build (i.e., SYSGEN) or boot an extended MPX-32 image on systems without this support result in a fatal abort.

Control between extended and nonextended MPX-32 code is performed by adaptive sequences. These sequences are generated by a combination of macro assembler directives and special communication sequences that are recognized by SYSGEN object processing. The adaptive sequences enable code linkage (e.g., branch requests) between extended and nonextended MPX-32 code. These sequences change the instruction mode so that code executed in extended MPX-32 is in base mode, and code executed in nonextended MPX-32 is in nonbase mode. Code linkages for extended to extended code or nonextended to nonextended code do not require adaptive sequences.

The Macro Assembler SSECT directive places code and data in the appropriate section of the operating system. Extended MPX-32 code is placed in the EXT_MPX section, adaptive code is placed in the ADP_MPX section, and DSECT data is placed in the DSECT section.

To allow direct reference by the extended code section of the split image, the communication region, the DSECT section, and the ADP_MPX section must be within the first 16KW of physical and logical memory.

Note: When using extended MPX, there is a minimal amount of code executed in the adaptive sequences to switch between non-base and base execution modes (and vice-versa). Generally, the time required for this mode switch is not perceptible to the user.

Additionally on CONCEPT 32/87 systems only, task context switch times may increase when extended MPX is used. Any increase is associated with the number of map image descriptors (which are pre-loaded by hardware at context switch time) associated with the task, including those describing extended MPX. Therefore, it is desirable to minimize this number (called the "map span"). This can be accomplished by using the EXTDMPX directive (whether in SYSGEN or CATALOG) and specifying either MINADDR (if sufficient logical task address space exists) or the lowest map block number where no address space conflicts will result.

Figure 3-2 shows physical memory of extended MPX-32.

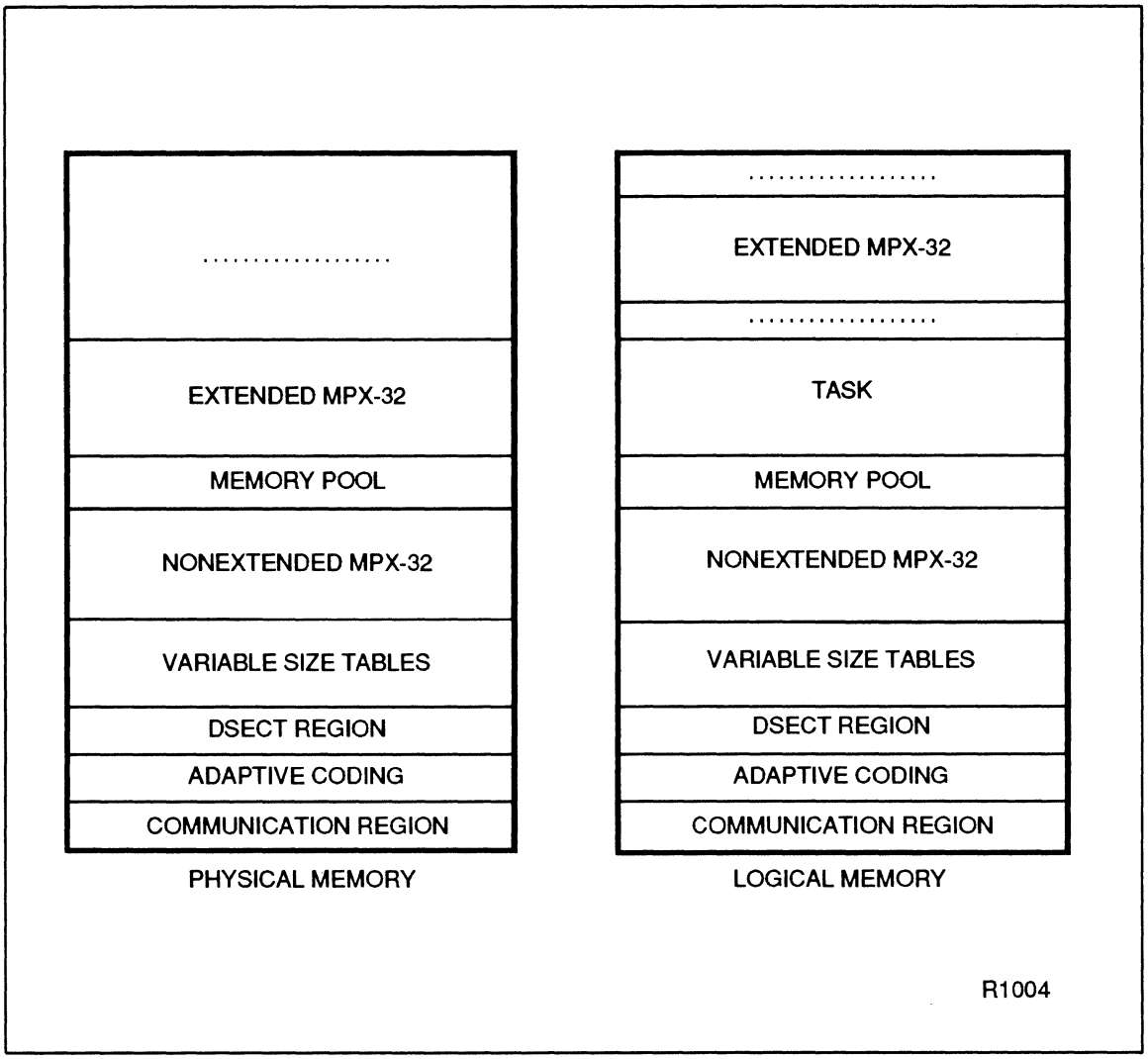


Figure 3-2
Extended MPX-32 Physical Memory

Extended MPX-32 (Expanded Execution Space)

Modules that can operate in extended MPX-32 are:

- Resource Allocator (H.ALOC)
- Executive Subroutine (H.EXSUB)
- File System Executive (H.FISE)
- Memory Management (H.MEMM)
- System Services (H.MONS)
- Program Trace (H.PTRAC)
- Resource Management (H.REMM)
- Resident Execution Services (H.REXS)
- Task Management (H.TAMM)
- Terminal Services (H.TSM)
- Volume Management (H.VOMM)

The object code for these modules is compressed into OH.32_E, a SYSGEN input object file.

For users who have modified these modules, or who want to move their own modules to extended memory, see the section How to Create an Extended MPX-32 System in this chapter for further details.

The Macro Assembler can generate extended and nonextended object code from the same properly converted source files. Assigning the logical file code PRE to MPX_EXT generates extended MPX-32 object, while assigning it to MPX_NON generates nonextended object.

Figure 3-3 illustrates program flow control for extended MPX-32.

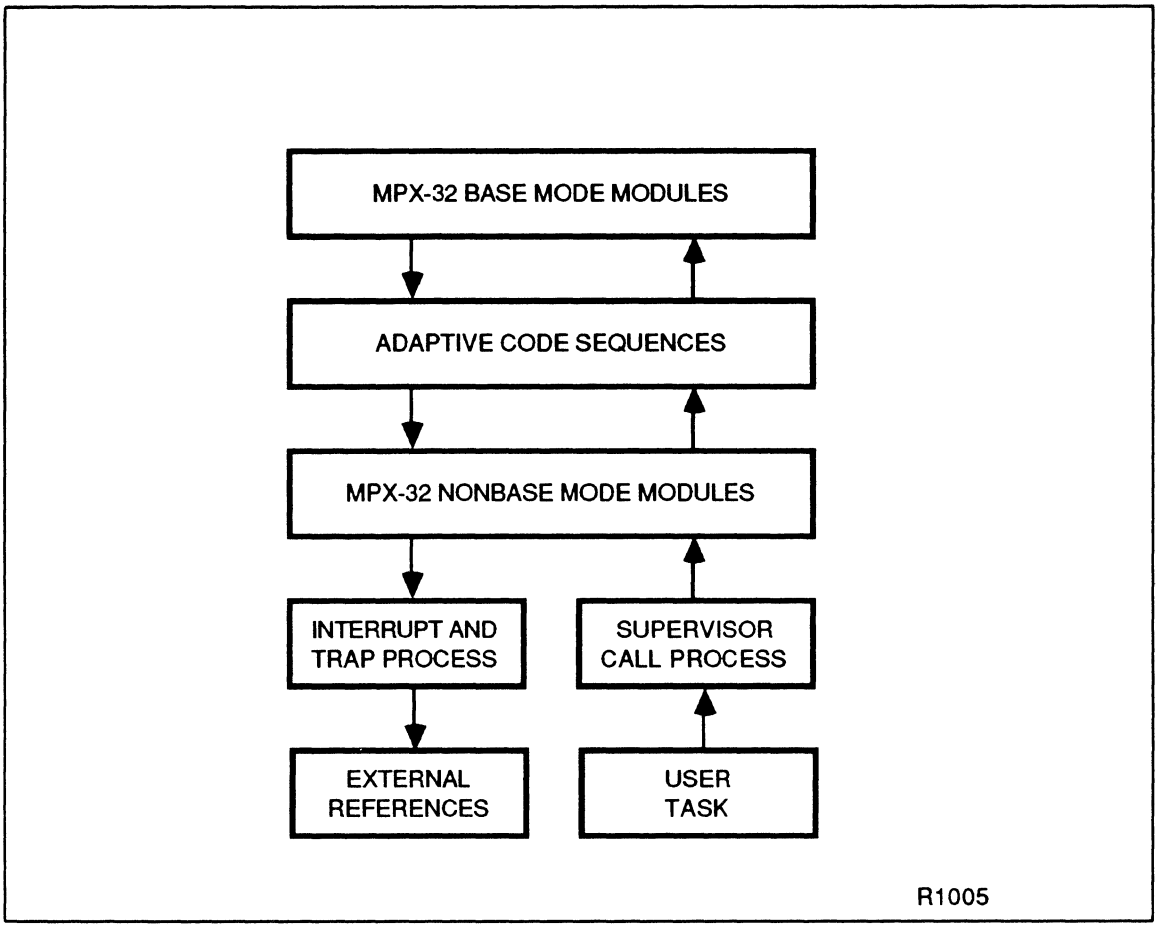


Figure 3-3
Extended MPX-32 Program Flow Control

3.10.1 SYSGEN Information for Extended MPX-32

When SYSGEN is activated, an object prescan is performed, followed by an object load. The object file assigned to logical file code OBR is scanned first. After this is completed, the object file assigned to the logical file code OBJ is scanned. This allows the extended MPX-32 object code in OH.32_E to preempt the equivalent nonextended version.

When creating an extended MPX-32 system, SYSGEN completes the following:

- SYSGEN performs two passes on the input object files. The first pass determines the size of the DSECT and adaptive sections. The second pass builds the system.
- SYSGEN resolves all base mode references when an image is created. The DSECT sections of the affected modules are placed in nonextended MPX-32, while the EXT_MPX sections are placed in extended MPX-32.
- SYSGEN provides the linkage required by the split image operating system to switch between the base and nonbase instruction modes. This additional adaptive code is placed in the ADP_MPX section which is located in the first 16KW of memory.
- SYSGEN resolves all references between the extended and nonextended modules.

3.10.2 SYSGEN EXTDMPX Directive

The SYSGEN EXTDMPX directive designates where extended MPX-32 is logically mapped into a task's address space. This directive can be overridden by the CATALOG and TSM EXTDMPX directives. If a parameter is not specified, the default is MINADDR.

The EXTDMPX directive can be used on a system wide basis to designate the starting address of the TSA and extended MPX-32 (if configured). This directive applies only to the TSA when extended MPX-32 is not configured.

The NOTSA or TSA option is ignored when the load module has been catologed in the compatible mode, or using the TSA keyword in the Cataloger EXTDMPX directive. The NOTSA or TSA option is effective only when the load module has been cataloged using the SYSTSA keyword in the Cataloger EXTDMPX directive. When this requirement is met, a TSM or M.PTSK request overrides the SYSGEN request.

If the EXTDMPX directive is not used, the default is MINADDR and NOTSA. This directive establishes the default TSA and extended MPX-32 logical starting address unless overridden via the CATALOG, TSM, or M.PTSK assignments.

Syntax

EXTDMPX={*logmapbl* | **MAXADDR** | **MINADDR**} [**,NOTSA** | **,TSA**]

logmapbl is a decimal value between 64 and 2047 that specifies a starting map block in the task's logical address space where the TSA (optionally) and extended MPX-32 (if configured) are positioned. The NOTSA or TSA keyword controls positioning of the TSA.

MAXADDR positions the TSA (optionally) and extended MPX-32 (if configured) at the top of the the task's logical memory. The NOTSA or TSA keyword controls positioning of the TSA.

MINADDR positions the TSA and extended MPX-32 (if configured) at the bottom of the task's logical memory above MPX-32 (when mapped in), and below the task's DSECT. The TSA keyword defaults to NOTSA for MINADDR.

NOTSA directs the logical position of the TSA to be above MPX-32 (when mapped in) and below extended MPX-32 (if configured and at MINADDR), and below the task's DSECT.

TSA directs the repositioning of the TSA in accordance with the MAXADDR, or *logmapbl* specification used. For MAXADDR the TSA is located at the top of the task's logical memory followed by extended MPX-32 (if configured). For *logmapbl*, the TSA logically starts at *logmapbl* followed by extended MPX-32 (if configured).

Note: An error, `***INVALID KEYWORD`, is displayed if `NOTSA` or `TSA` keywords are incorrectly spelled and the image is not built.

At runtime, values for `MAXADDR` or *logmapbl* that conflict with the task's code, data, or partition memory requirements cause an abnormal termination in the task activation.

3.10.3 SYSGEN Aborts and Errors for Extended MPX-32

If extended MPX-32 errors are detected during SYSGEN, the following aborts may be generated:

Code	Message
SG37	COMMUNICATION REGION + DSECT + ADAPTIVE REGION EXCEEDS 16KW
SG38	MPX EXTENDED CODE AREA EXTENDS PAST LOGICAL LIMIT
SG39	INVALID MPX EXTENDED CODE AREA LOGICAL MAP START
SG98	ERROR ENCOUNTERED DURING OBJECT PROCESSING PRECEDED BY MESSAGE DESCRIBING THE ERROR CONDITION

3.10.4 How to Create an Extended MPX-32 System

The following information describes how to create an extended MPX-32 system using the default modules and/or user-created modules.

The MPX-32 SDT contains an input object file, OH.32_E. This object file contains the modules that can execute in extended MPX-32. These modules are:

- Resource Allocator (H.ALOC)
- Executive Subroutine (H.EXSUB)
- File System Executive (H.FISE)
- Memory Management (H.MEMM)
- System Services (H.MONS)
- Program Trace (H.PTRAC)
- Resource Management (H.REMM)
- Resident Execution Services (H.REXS)
- Task Management (H.TAMM)
- Terminal Services (H.TSM)
- Volume Management (H.VOMM)

To create a split image system with any combination of the modules and any user-created extended MPX-32 modules, complete steps 1 to 6 listed below.

To create a split image system with these modules in extended MPX-32, complete steps 3 to 6 listed below.

1. Edit JH.32_E, the input source file, to contain the modules that will execute in the extended MPX-32 region. These modules must meet the extended MPX-32 programming considerations.
2. Use COMPRESS to compress JH.32_E into OH.32_E, the input object file. COMP32, an SDT file, contains the JCL for compressing nonextended modules that are listed in JH.32 into OH.32, and extended modules that are listed in JH.32_E into OH.32_E.
3. Assign OH.32 to logical file code OBJ.
4. Assign OH.32_E, the compressed input object file, to logical file code OBR. If OH.32_E is not assigned to OBR, the system cannot create a split image and extended MPX-32 is not available.
5. Specify the SYSGEN EXTDMPX directive (optional). This directive specifies the starting logical map block number of extended MPX-32. If EXTDMPX is not specified, extended MPX-32 is mapped in the end of the task service area (MINADDR).

Note: If EXTDMPX=MINADDR, some languages and utilities (e.g. FORTRAN, Macro Assembler) may not load (RM65), or may not be able to allocate dynamic executable memory using M.GD, an M.MPXMAC macro, or X:GDSPCE, a FORTRAN 77+ subroutine. To avoid this, use EXTDMPX=MAXADDR at catalog time or in the JCL.

6. Run SYSGEN.

3.10.5 How to Relocate Extended MPX-32

For tasks not affected by the size of the operating system, MINADDR is the compatible mode of operation. For tasks impacted by the size of the operating system, it is necessary to override the SYSGENed logical position of extended MPX-32. The procedures for this are:

- Cataloger EXTDMPX directive
- TSM EXTDMPX directive

Figure 3-4 shows the logical task areas that can be created using an EXTDMPX directive. Descriptions for the EXTDMPX directives follow the figures.

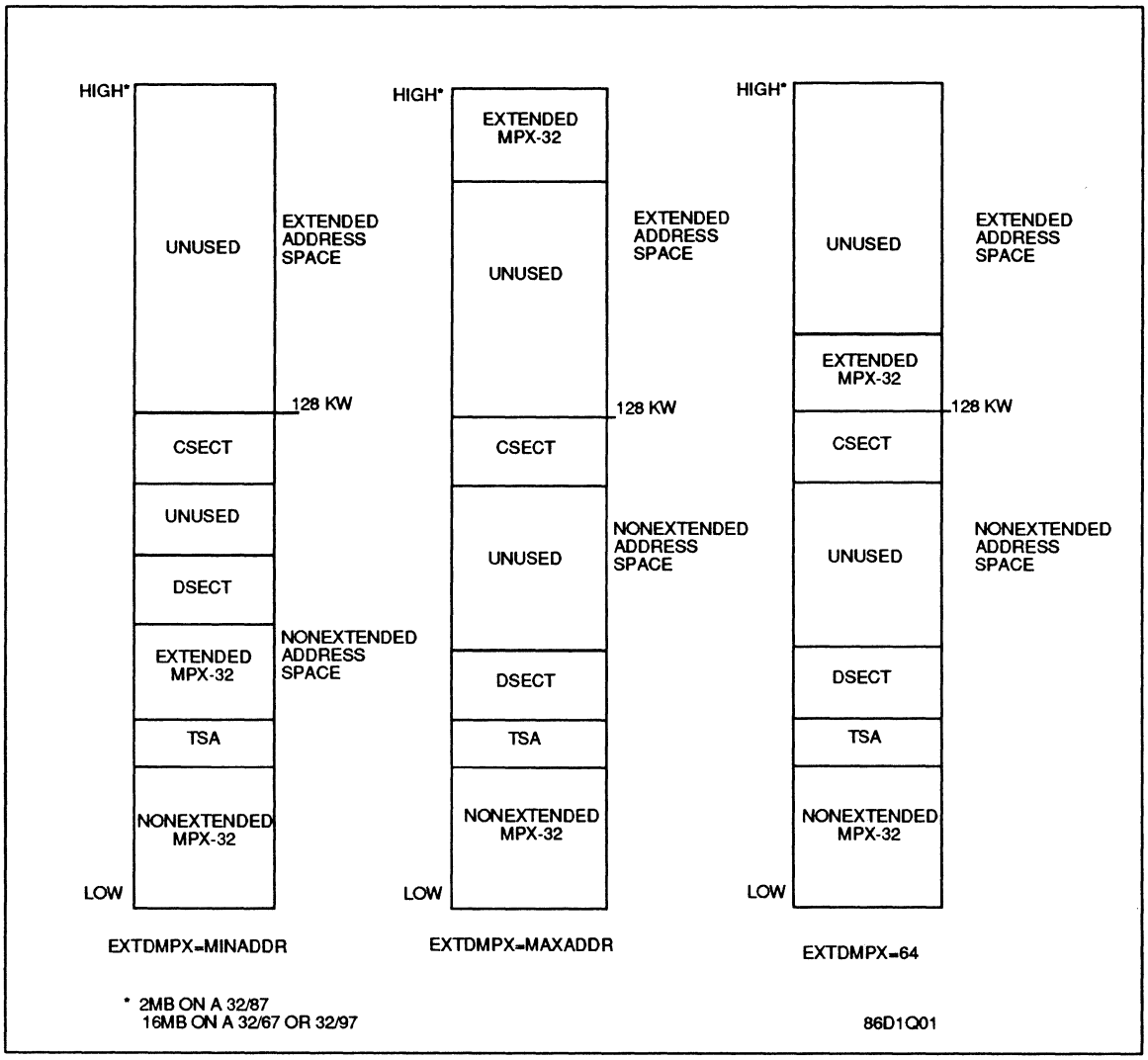


Figure 3-4
Tasks' Logical Address Space Using Extended MPX-32

3.10.6 CATALOG EXTDMPX Directive

This directive designates where extended MPX-32 is mapped into the task's logical address space when the task executes. This directive may be used to dynamically override the SYSGEN EXTDMPX directive. If this directive is not specified when building a load module, the SYSGEN EXTDMPX directive remains in effect when the task is executed.

The EXTDMPX directive can also be used to control repositioning the task's TSA in the task's logical address space. When extended MPX-32 is configured, the EXTDMPX directive positions the TSA and extended MPX-32 in the logical address space of the task being cataloged. When extended MPX-32 is not configured, this directive applies only to the TSA. This directive is functionally identical to the EXTDMPX directive used in previous releases of MPX-32 and the Utilities unless the optional TSA or SYSTSA keyword is used. Existing load modules that were cataloged using previous versions of the Utilitites Release 3.2 run in the compatible NOTSA mode. The NOTSA, TSA, and SYSTSA keywords are mutually exclusive.

Note: If EXTDMPX=MINADDR, some languages and utilities (i.e. FORTRAN, Macro Assembler) may not load (RM65), or may not be able to allocate dynamic executable memory using M.GD, an M.MPXMAC macro, or X:GDSPCE, a FORTRAN 77+ subroutine. To avoid this, use EXTDMPX=MAXADDR at catalog time or within the JCL.

Syntax

EXTDMPX[=] { *logmapbl* | **MINA**DDR | **MAXA**DDR } [,**NOTS**A | ,**TSA** | ,**SYST**SA]

logmapbl is a decimal value between 64 and 2047 that specifies a starting map block in the task's logical address space where TSA (optionally) and extended MPX-32 (if configured) are positioned. The NOTSA, TSA, or SYSTSA keyword controls positioning of the TSA.

MINADDR positions the TSA and extended MPX-32 (if configured) at the bottom of the task's logical memory above MPX-32 (when mapped in), and below the task's DSECT. The TSA keyword defaults to NOTSA for MINADDR.

MAXADDR positions the TSA (optionally) and extended MPX-32 (if configured) at the top of the task's logical memory. The NOTSA or TSA keyword controls positioning of the TSA.

NOTSA directs the logical position of the TSA to be above MPX-32 (when mapped in) and below extended MPX-32 (if configured and at MINADDR), and below the task's DSECT.

TSA directs the repositioning of the TSA in accordance with the MAXADDR, or *logmapbl* specification used. For MAXADDR the TSA followed by extended MPX-32 is located at the top of the task's logical memory respectively. For *logmapbl*, the TSA followed by extended MPX-32 (if configured), logically starts at *logmapbl*.

SYSTSA defers positioning the TSA and extended MPX-32 (if configured) until runtime. At runtime the TSM, M.PTSK, or the SYSGEN specification directs positioning of the TSA and extended MPX-32.

3.10.7 TSM EXTDMPX Directive

This directive dynamically overrides the SYSGEN and CATALOG directives for the logical starting address of extended MPX-32. This directive is ignored if the task is multicopy shared, or is a base mode task.

Note: If EXTDMPX=MINADDR, some languages and utilities (i.e. FORTRAN, Macro Assembler) may not load (RM65), or may not be able to allocate dynamic executable memory using M.GD, an M.MPXMAC macro, or X:GDSPCE, a FORTRAN 77+ subroutine. To avoid this, use EXTDMPX=MAXADDR at catalog time or within the JCL.

The NOTSA or TSA option is ignored when the load module has been catologed in the compatible mode, or using the TSA keyword in the Cataloger EXTDMPX directive. The NOTSA or TSA option is effective only when the load module has been cataloged using the SYSTSA keyword in the Cataloger EXTDMPX directive. When this requirement has been met, a TSM or M.PTSK request will override the SYSGEN request.

Syntax

EXTDMPX[=] {*logmapbl* | **MAX**ADDR | **MIN**ADDR} [,**NOT**SA | **TSA**]

logmapbl is a decimal value between 64 and 2047 that specifies a starting map block in the task's logical address space where the TSA (optionally) and extended MPX-32 (if configured) are positioned. The NOTSA or TSA keyword controls positioning of the TSA.

MAXADDR positions the TSA (optionally) and extended MPX-32 (if configured) at the top of the the task's logical memory. The NOTSA or TSA keyword controls positioning of the TSA.

MINADDR positions the TSA and extended MPX-32 (if configured) at the bottom of the task's logical memory above MPX-32 (when mapped in), and below the task's DSECT. The TSA keyword defaults to NOTSA for MINADDR.

NOTSA directs the logical position of the TSA to be above MPX-32 (when mapped in) and below extended MPX-32 (if configured and at MINADDR), and below the task's DSECT.

TSA directs the repositioning of the TSA in accordance with the MAXADDR, or *logmapbl* specification used. For MAXADDR the TSA followed by extended MPX-32 (if configured) is located at the top of the task's logical memory. For *logmapbl*, the TSA followed by extended MPX-32 (if configured) logically starts at *logmapbl*.

Note: An error, `*ILLEGAL ENTRY`, is displayed on the user's terminal if `NOTSA` or `TSA` keywords are incorrectly spelled.

At runtime, values for `MAXADDR` or *logmapbl* that conflict with the task's code, data, or partition memory requirements cause an abnormal termination in the task activation.

3.11 Extended TSA (Expanded Execution Space)

The extended TSA feature is optionally available to any nonbase task. This feature allows the user to move the task's TSA into the task's indexed (extended) address space. Positioning the TSA in the task's indexed address space results in more direct executable address space for code and directly addressable data.

The TSA varies in size from task to task and is greatly increased by the number and type of I/O resources required by the task. Moving the TSA increases the user task's direct address space by a minimum of 2 map blocks for tasks with little I/O requirements and many more map blocks for tasks with heavy I/O requirements. There is no degradation in performance by moving the task's TSA.

The position of the task's TSA is determined during task loading time and is fixed for the duration of the task. Each task may choose the position of its TSA.

The EXTDMPX directive has added the optional keywords NOTSA (do not move the TSA) and TSA (move the TSA) to direct MPX-32 in positioning the task's TSA. Since the user may need to modify source in order to take advantage of this feature, the CATALOG EXTDMPX directive is the central point of control for initiating the move TSA capability. The SYSTSA keyword enables the TSM, M.PTSK, and SYSGEN specifications for moving the TSA. The user can position the TSA by using one of the following means:

CATALOG EXTDMPX directive

TSM EXTDMPX directive

M.PTSK SVC call

SYSGEN EXTDMPX directive

Tasks cataloged without the NOTSA, TSA, or SYSTSA keywords on the CATALOG EXTDMPX directive will have their TSA and extended MPX-32 positioned compatibly with versions of MPX-32 prior to 3.5. When a task is cataloged using the TSA or NOTSA option on the CATALOG EXTDMPX directive any TSM, M.PTSK, or SYSGEN request is ignored. When the task is cataloged specifying the SYSTSA option of the CATALOG EXTDMPX directive, the position of the TSA and extended MPX-32 (if present) may be determined by any one of the following requests (listed in order of precedence): TSM, M.PTSK, or SYSGEN EXTDMPX.

When the task is executing on a split MPX-32 image, the extended section of MPX-32 is positioned logically above and contiguous with the TSA. The TSA is positioned at the logical address specified by the EXTDMPX directive keyword; MINADDR, MAXADDR, or map block number. On non-split MPX-32 images only the TSA is applicable.

3.11.1 Relocating the TSA

User modules that reside within MPX-32 and do not reference data within the TSA of non-resident tasks do not require code changes.

User modules that reside within MPX-32 and reference data within the TSA of non-resident tasks may require code changes.

- When the non-resident task's TSA is repositioned at other than MINADDR, extended addressing must be set to reference data within that task's TSA.
- Resident modules using C.TSAD to obtain the logical start of the non-resident task's TSA must now use the M.TSAD macro to obtain the task's TSA address when it is positioned at other than MINADDR.

For tasks not impacted by the size of their TSA, the default NOTSA keyword is the compatible mode of operation.

Tasks that are impacted by the size of their TSA and reference data structures within their TSA may require some code changes before repositioning their TSA.

- When the TSA is positioned at other than MINADDR, the task must set extended addressing to reference data within its TSA.
- Tasks using C.TSAD to obtain the logical start to their TSA must now use the M.GTSAD system service call to obtain their TSA address when it is positioned at other than MINADDR.

Figure 3-5 shows the logical task areas that can be created using the EXTDMPX directive with the TSA keyword. Refer to the appropriate manual for further descriptions concerning the following: the EXTDMPX directive, refer to Reference Manual Volume II; the M.TSAD macro, refer to Technical Manual Volume II; and the M.GTSAD SVC call, refer to Reference Manual Volume I.

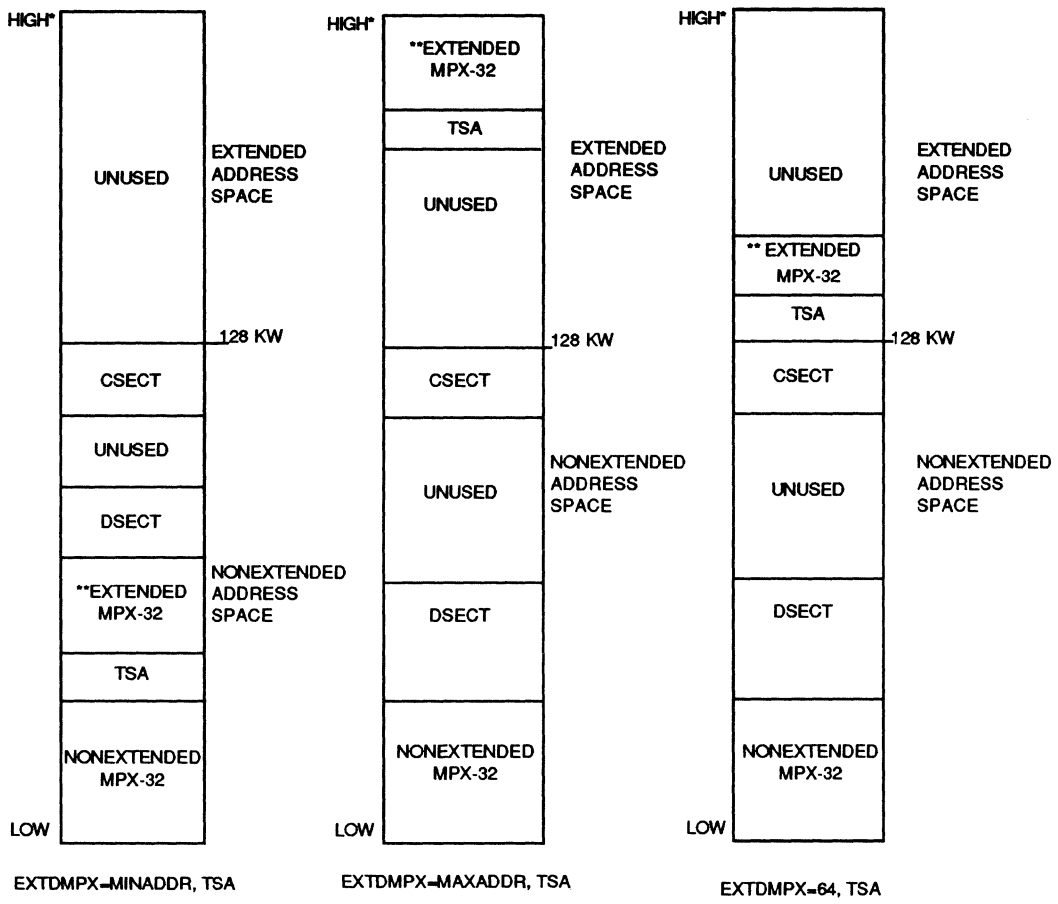


Figure 3-5
Task's Logical Address Space Using the EXTDMPX Directive with TSA Keyword

3.12 Mapped Out Option (CONCEPT 32/2000 Only)

The mapped out feature is optionally available to any nonbase task executing on a CONCEPT 32/2000 processor running an MPX-32 mapped out system image. The creation of a mapped out MPX-32 image is specified at SYSGEN time via the MACHINE directive or by an explicit assignment of LFC OBJ to a compressed mapped out object file. This feature is available on a task by task basis and allows the task to execute with MPX-32 removed from the task's logical address space. Executing a task in the mapped out mode results in more direct (execution) address space for code and directly addressable data with no performance degradation.

Since the user may need to modify source in order to take advantage of this feature, the CATALOG ENVIRONMENT directive is the central point of control for initiating the mapped out capability. It has the NOMAPOUT and MAPOUT keywords added to explicitly define a task's mapped out state as well as the SYSMAP keyword that defers the request until runtime. The user can specify the mapped out or mapped in options using :

CATALOG ENVIRONMENT directive

TSM MAPOUT or NOMAPOUT directives

M.PTSK SVC call

SYSGEN MAPOUT or NOMAPOUT directives

Load modules built with a previous version of the MPX-32 Utilities Release 3.2 Cataloger run in the compatible mode with MPX-32 mapped into their address space. Load modules built without the NOMAPOUT, MAPOUT, or SYSMAP keywords on the CATALOG ENVIRONMENT directive will run in the compatible mode. When a task is cataloged using the MAPOUT or NOMAPOUT option on the CATALOG ENVIRONMENT directive any TSM, M.PTSK, or SYSGEN request is ignored. When the task is Cataloged with the SYSMAP option on the CATALOG ENVIRONMENT directive, the mapped out or mapped in option may be determined by any one of the following requests (listed in order of precedence): TSM, M.PTSK, or SYSGEN MAPOUT or NOMAPOUT. Attempts to execute a mapped out task on other than a mapped out MPX-32 system image will be ignored and the task will execute mapped in.

The mapped state of MPX-32 with respect to the task is determined during task loading time and is fixed for the duration of the task.

4 Volume Resource Management

4.1 Symbolic Resource Management

The MPX-32 operating system manages all of its major resources with a resource management scheme. With this scheme, resources are named symbolically in directories. The directory entries establish the symbolic name to a physical resource relationship.

Each directory entry for a resource points to a resource descriptor. These resource descriptors are the central data structure for volume resource management. They contain all information required to manage the resource. They are stored on disk and some portions are brought into memory as needed for efficiency. Utilizing this data structure, all primary resources are managed in the same way.

The symbolic directory entry and the resource descriptor are defined when the resource is created. A resource descriptor (RD) contains a unique identifier that can identify the resource internally. The resource identifier (RID) allows the name to be expressed and used by the system in a short and unambiguous manner.

The MPX-32 operating system also provides mechanisms that address the problems of static and dynamic resource management.

Static resource management defines the access, protection, and allocation attributes of a resource. To handle static management considerations adequately, all attributes are specifiable when the resource is defined (created).

Static resource management provides guarantees to tasks that are activated (i.e., requested for execution) ensuring the required resources will be available when the task begins execution.

Dynamic resource management provides methods that allow convenient dynamic attachment and access to a resource. To handle this requirement, the operating system provides methods to enable a task to attach or access a resource, to enqueue on a currently unavailable resource, to gain exclusive access to a shared resource, and to synchronize on the use of a shared resource.

This generalized resource management scheme means the same protection mechanism, the same resource management algorithms, and other resource operations work in a standard and predictable manner on all resources.

This general scheme does not imply users of the resources cannot detect differences between the types of resources but that, with the exception of memory attachment, the user may not have to be concerned with the differences. For users having specific needs or requirements for certain types of resources, a standard resource inquiry mechanism is available to report all information known about a resource.

4.1.1 Types of Resources

There are four major types of resources managed by the MPX-32 operating system: disk volumes, directories, files, and memory partitions.

4.1.2 Classes of Resources

There are two classes of resources managed by the operating system: shareable and nonshareable.

Shareable resources can be accessed by two or more concurrently executing tasks. For example, the resource can be attached to a task while another task is currently attached to it. Any major resource of the system can be declared as shareable when the resource is created; for example, defined to the system.

Nonshareable resources can only be used by a single task at any time. For example, the resource cannot be attached to a task while another task is currently attached to it. Disk volumes, directories and files can be declared as nonshareable when the resource is created; for example, defined to the system.

4.1.3 Classes of Resource Users

The MPX-32 operating system divides the users of resources into three classes: the owner of the resource, any member of a group of users of the resource, and any other arbitrary user of the resource. These classes control access rights to the resource.

- resource owners — The power to control access rights to the resource is given to the owner of the resource. The resource owner is determined at the time the resource is created. Generally, the resource owner is the creator of the resource. When the resource is created, the owner assigns the logical attributes and protection for the resource. These attributes can be changed after the resource is created, but this privilege is allowed only to the resource owner.
- resource project groups — When a resource is created, the creator of the resource can define the name of a group of users and specify the resource's attributes and protection as it applies to all members of that group.
- other resource users — A user who is not the owner or a member of the project group associated with the resource is also given a set of resource attributes and protection as it applies within this perspective.

4.1.4 Shareable Resource Control Mechanisms

Shareable resources can be attached and accessed in three ways: exclusive, implicitly shared, and explicitly shared. These three ways are mutually exclusive of one another.

- exclusive use — When a shareable resource is attached for exclusive use, the resource is not available for use by any other task until the resource is detached from the using task.
- implicit shared use — When a shareable resource is attached for implicitly shared use, the resource can only be attached by another task that is attaching the resource in a compatible access mode.
- explicit shared use — When a shareable resource is attached for explicit sharing, the resource can only be attached by another task attempting to attach the resource for explicit sharing. Tasks that are explicitly sharing a resource do not have to access the resource in compatible access modes but are expected to use the shared resource control mechanisms designed to preserve the integrity of the resource. See the Sharing Files section of this chapter. Memory partitions are always attached for explicit shared use.

4.2 General Resource Control

The needs for resource control include the following:

- time-critical task must be able to quickly attach and access a resource
- task must be able to enqueue on the access or attachment of a resource
- task must be able to gain exclusive use of a shared resource
- task that is sharing a resource must be able to synchronize on the use of that resource

Potential users of resources must consider that conflicts for use of resources can occur. When conflicts occur, the operating system furnishes the user with mechanisms that aid in resolving the conflicts. These mechanisms are:

- the ability to attach a resource statically or dynamically
- the ability to enqueue for attachment or access to a resource
- the ability to attach a shareable resource for exclusive use
- the ability to synchronize on the use of an attached shareable resource

4.2.1 Enqueue and Synchronous Notification Mechanism

When a task is attempting to dynamically attach a resource, the resource may not be available. If a denial return is furnished, the task is notified immediately and control is returned to the task. If a denial return is not furnished, the task is enqueued for the resource and removed from execution until the resource becomes available.

With the automatic enqueue mechanism, the task can optionally specify the length of time it is willing to wait for availability of the resource. If the resource is available within the prescribed time, the task is given normal completion status. If the resource is not available in time, abnormal completion status is reported to the task.

4.2.2 Dequeue Mechanism

When an unavailable resource is released, the highest priority task enqueued for the resource is attached to the resource and is made eligible for execution.

4.3 Shareable Resource Access Control

After a shareable resource is attached, the tasks using the resource need mechanisms for controlling its access. The following mechanisms provide lock control and enqueue capabilities at attachment and momentary synchronization lock control and enqueue capabilities at access.

4.3.1 Shareable Resource Locking

When a task desires to gain exclusive access to a shareable resource, the task can attach to the resource for exclusive use. Once attached, no other tasks in the system can attach to the resource. The resource remains attached until released.

Once a resource is attached, subsequent requestors of the resource may enqueue for the resource.

4.3.2 Shareable Resource Synchronization

If a set of tasks desire to synchronize access to a shareable resource, the tasks may all attach the resource for explicit shared usage and then employ the resource synchronization locking mechanism. A resource that is synchronization locked should not be accessed by any task other than those that have secured the synchronous lock, but the operating system will not prevent concurrent access by other tasks not using the lock.

Once a resource is locked, subsequent requestors of the lock may enqueue for the lock.

4.4 Standard Disk Structure

A uniform volume structure, coupled with a two-level directory structure, is implemented in the MPX-32 operating system. The operating system supports removable disk pack volumes. These volumes are constructed and managed using a standard format. See the Volumes section of this chapter.

4.4.1 Directory Structure

The operating system incorporates a two-level directory structure that gives each user of the system a perspective of being the only user of the system. See the Directories section in this chapter.

The directory structure and the overall volume organization provide fast access for the time-critical user. Specifically, a directory entry always points to a resource descriptor. The resource descriptor has associated with it a unique volume relative address. This reduces all symbolic file names to a short unambiguous name, for example, the file identifier. By furnishing the file identifier, time-critical tasks are guaranteed to acquire the file description in one access.

4.4.2 Root Directory

Each removable volume contains a master directory referred to as the root directory. This directory is the directory of all directories defined on a volume. The root directory can also contain the definition of other resources which are not directories. All named resources on the volume can be located through the root directory.

4.4.3 Current Working Directory

A current working directory is a directory located on a mounted volume. A task can reference resources defined in the directory without specifying a complete pathname (see the following pathname discussion). A task can specify a 1- to 16-character resource name and the operating system will prefix the current working directory and volume name to form a complete pathname to locate the resource.

In the interactive environment, a current working directory is associated at logon.

In the batch environment, a current working directory is associated when a job starts execution.

In the real-time (independent) environment, the current working directory is the same as the one associated with the activator.

All named resources on a volume are accessible from the root directory.

4.5 Pathnames

A pathname is an ASCII character string used to refer to any named resource defined on a volume. The pathname contains a sequence of symbolic names. Each symbolic name is delineated by special characters embedded in the pathname string. The special characters enable the operating system to directly interpret the meaning of each symbolic name. The last symbolic name in the pathname string is the target of the pathname. A pathname target can be a file, directory or memory partition.

With a pathname, any named resource existing on a volume can be located for attachment or inquiry. To locate a resource, the operating system requires the identity of the volume and directory containing the resource (target). As previously mentioned, the special characters embedded in the pathname string uniquely identify the required components. These required components can be specified or implied.

A pathname that contains each of the required components is referred to as a fully qualified pathname. Fully qualified pathnames are processed by examining the pathname string from left to right or the end of the string. As each special character is detected, the identified component is located in the appropriate system structure. An identified volume is found in the mounted volume table (MVT), a directory is found in the root directory of the volume and the target resource is found in the specified directory. If a pathname component is not found, the processing of the pathname is terminated at that point.

Each active task in the operating system has an associated current working volume and directory. The current working volume and directory are associated with the task when it becomes active in the operating system. The current working volume and directory allow the task to reference resources defined in this association by implication. That is, the task can reference a resource with a pathname that is not fully qualified. For example, such a pathname may indicate the pathname execution is to start at the root directory. In this case, the current working volume is implied. As another example, only the resource name is specified. In this case, the current working volume and directory are implied. The task can view this operation as if the operating system were supplying the missing pathname components. The task must consider the operating system is concatenating the current working volume or current working volume and directory to the pathname supplied by the task. This concatenated string must resolve to a fully qualified pathname to be successfully executed by the operating system.

4.5.1 Executing Pathnames

When a pathname is presented to a command or service, the operating system parses the pathname. As mentioned previously, the operating system interprets special characters embedded in the pathname to have a specific meaning.

Special characters that are allowed to be specified in a pathname are as follows:

Special Character	Description
@	a volume
^	root directory of a volume
(	named directory on a volume
)	named resource on a volume

4.5.2 Fully Qualified Pathnames

A fully qualified pathname consists of all the information the operating system requires to locate and identify a resource. Whenever the following pathnames are presented to the log directive or service, only the specified resource will be logged.

1. *@volume^(directory)resource*

   @ indicates a volume is being specified

   volume is the 1- to 16-character name of the volume where *resource* resides. *volume* must be physically mounted on the system. The system determines the physical device where *volume* is mounted.

   ^ indicates the root directory of the volume is being specified

   (indicates a directory is being specified

   directory is the 1- to 16-character name of the directory on *volume* in which *resource* is defined

) indicates a resource is being specified

   resource is the 1- to 16-character name of the resource to be located in *directory* on *volume*

2. *@volume(directory)resource*

   @ indicates a volume is being specified

   volume is the 1- to 16-character name of the volume where *resource* resides

   (indicates a directory is being specified

   directory is the 1- to 16-character name of the directory on *volume* where *resource* is defined

   In this example, the optional special character to indicate the root directory is not used. However, this pathname is equivalent to the preceding example.

) indicates a resource is being specified

   resource is the 1- to 16-character name of the resource to be located in *directory* on *volume*

3. *@volume^resource*

@	indicates a volume is being specified
volume	is the 1- to 16-character name of the volume where *resource* resides
^	indicates the root directory of the volume is being specified
resource	is the 1- to 16-character name of the resource to be located in the root directory on *volume*

4.5.3 Partially Qualified Pathnames

The operating system allows the use of pathnames that are not fully qualified. The use of such a partially qualified pathname causes the operating system to substitute symbolic names that are not directly specified when the pathname is presented to a directive or service.

The operating system makes the appropriate substitutions based on the association with the user's current working volume and current working directory.

1. *^(directory)resource*

^	indicates the root directory of the current working volume
(	indicates a directory is being specified
directory	is the 1- to 16-character name of the directory on the volume where *resource* is defined
)	indicates a resource is being specified
resource	is the 1- to 16-character name of the resource to be located in *directory* on *volume*

2. *^resource*

^	indicates the root directory of the current working volume
resource	is the 1- to 16-character name of the resource defined in the root directory of the current working volume

4.5.4 Fully Qualified Pathnames for Directories Only

Any of the preceding pathnames can be used to locate or reference directories or any other resources defined on a volume. The following pathname formats can only be used when referencing directories. These formats are only allowed in the change directory and log resource commands and services.

1. *@volume^(directory)*
   @volume(directory)

@	indicates a volume is being specified
volume	is the 1- to 16-character name of the volume where *directory* resides
^	indicates the root directory of the volume is being specified
(	indicates a directory is being specified
directory	is the 1- to 16-character name of the directory to be located in *volume*

   Usage:

   These pathname formats are equivalent to each other. If this pathname format is used with a LOG RESOURCE directive, all files in *directory* on *volume* are logged.

2. *@volume^directory*

@	indicates a volume is being specified
volume	is the 1- to 16-character name of the volume where *directory* resides
^	indicates the root directory of *volume* is being specified
directory	is the 1- to 16-character name of the directory to be located in *volume*

Usage:

If this pathname format is used with the CHANGE DIRECTORY directive, the directory name is changed to *directory* on *volume*.

If this pathname format is used with the LOG RESOURCE directive, *directory* on *volume* is logged.

4.5.5 Partially Qualified Directory Pathnames

The operating system allows the use of pathnames that are not fully qualified. The use of such a partially qualified pathname causes the operating system to substitute symbolic names that are not directly specified when the pathname is presented to a command or service.

1. ^(*directory*)

^	indicates the root directory of the current working volume
(	indicates a directory is being specified
directory	is the 1- to 16-character name of the directory to be located on the current working volume

Usage:

If this pathname format is used with the CHANGE DIRECTORY directive, the directory would change to *directory*. If this pathname format is used with the LOG RESOURCE directory, all files in *directory* are logged.

2. *@volume^*

@	indicates a volume is being specified
volume	is the 1- to 16-character name of the volume where the required directory resides
^	indicates the root directory of the volume is being specified

   Usage:

   If this pathname format is used with the LOG RESOURCE directive and the ROOT= option is reset, all directories in *volume* would be logged. If this pathname format is used with the LOG RESOURCE directive and the ROOT= option is set, the root directory in *volume* would be logged. This pathname format is not valid with the CHANGE DIRECTORY directive.

3. ^

^	indicates the root directory of the current working volume. Most often, this form of pathname determines the names of all directories defined on the current working volume.

   Usage:

   If this pathname format is used with the LOG RESOURCE directive, all directories in the root directory of the current working volume are logged. This pathname format is not valid with the CHANGE DIRECTORY directive.

4. *^directory*

^	indicates the root directory of the current working volume
directory	is the 1- to 16-character name of the directory which resides on the current working volume

   Usage:

   If this pathname format is used with the CHANGE DIRECTORY directive, the default working directory is changed to the directory specified on the current working volume. If this pathname format is used with the LOG RESOURCE directive, the directory is logged.

4.6 Resource Protection

Protection is supplied for environments where protection is desired. When a resource is created, the user can specify the protection attributes of the resource. Since protection can be harmful as well as helpful, the user is advised to only protect resources to the appropriate level required. Each resource defined to the system has protection attributes that are unique and appropriate for the resource. For example, files are protected from being accessed in certain modes (see the File Access Modes section in this chapter), directories are protected from being searched or modified (see the Protecting Directories section in this chapter), and memory partitions are protected similarly to files (see the Protecting Memory Partitions section in this chapter).

Any of the major resources managed by the operating system can be protected from the perspective of the owner of the resource, a member of a group of users of the resource, or an arbitrary user of the resource. This protection scheme is versatile and gives the owner of the resource a reasonable means to control the resource.

When a resource is created, the process requesting the creation of the resource is the resource owner unless otherwise explicitly stated at the time of creation of the resource. The owner of a resource assigns the attributes of all the levels of protection.

4.7 System Administration

With MPX-32, a system administrator (SA) can be designated to control certain aspects of the operating system.

The SA defines who is allowed to logon the system in the key (M.KEY) file. Persons defined in this file are referred to as users. Each user logging on the system has an associated name, referred to as the owner name. While using the system, the user creates temporary or permanent resources. The owner's name is recorded in these resources to indicate their origin and to determine who is controlling the resources. Furthermore, the SA assigns in the M.KEY file the capabilities associated with each user.

The users allowed to logon the system are associated with a project group. The SA defines the projects in the project (M.PRJCT) file. There is not a rigid relationship between the users of the system and the projects to which they belong. Simply, each user has an associated project group at log on time. Once logged on, users can change their project group to any name contained in the project file. In some cases, the user must supply a key to have the project group changed. This key is associated with the project group name. Management of the project file is the responsibility of the SA and the system allows only tasks with the SA attribute to modify the key and project files.

The key and project files are optional. If these files are not present in the system configuration, any user can log on the system and can be a member of any project group.

Note: The SA is not restricted by any mechanism in the system. The SA can gain access to protected resources, can execute privileged system functions, etc.

4.8 Volumes

A volume is a formatted or unformatted storage medium that holds resources (files, directories, memory partitions) which can be accessed by name. A formatted volume is a disk medium with a standard MPX-32 format. Its resources can be protected through the formatting process or through the mount process.

4.8.1 Overview of Formatted Volumes

MPX-32 distinguishes three types of formatted volumes: system, user, and multiprocessor. Volume type is determined when the volume is mounted by the options specified in the mount request. The system volume is automatically mounted at system initialization, transparent to the user. Any volume mounted after system initialization is either a user or multiprocessor volume. The exception is the swap volume, which is a special-purpose user volume that is mounted on request with the system volume during system initialization.

Each volume must be physically mounted on a device before tasks can assign and access its resources. When requesting the physical mount of a volume, the requestor assigns an access attribute to the volume which designates the volume as public or nonpublic. This attribute determines how all subsequent tasks mount and dismount the volume and access its resources. Once physically mounted, public volume resources are available for immediate access to any task. Nonpublic volume resources are available to a task only after the task performs a logical mount of the volume.

Logical mount attaches a task to a nonpublic volume (public volumes do not require a logical mount) and establishes the task as a user of the volume's resources. A request for logical mount is either explicit or implicit. An explicit request is made by a task or user through a mount directive, system service call, or RRS entry supplied during task activation with the M.PTSK service (M_PTSK). An implicit request is issued automatically as the result of TSM task activation or a physical mount request for a nonpublic volume. Once logically mounted to a volume, a task or user remains mounted for the duration of a job or interactive session.

When a task no longer requires the use of a volume, it detaches through a logical dismount and, optionally, a physical dismount of the volume. A logical dismount detaches the requesting task from the volume. Logical dismount is requested explicitly through a dismount directive or a system service call. It is requested implicitly through task exit, end of job, or a TSM $EXIT command.

Physical dismount detaches a volume physically from the device where it is mounted. Physical dismount is explicitly requested by a task or user through a dismount command or a system service call. If other tasks or users are active on the volume, physical dismount is delayed, but all subsequent requests to mount the volume are denied. Use of resources on a public volume with dismount pending is not denied.

When the last task or user detaches from the volume, the volume is physically dismounted from the device.

4.8.2 Formatted Volume Type

There are three types of formatted volumes on MPX-32: system, user, and multiprocessor. On any MPX-32 system, there is only one system volume. All other volumes on the system are user or multiprocessor volumes.

4.8.2.1 System Volume

The system volume is the volume automatically mounted at system IPL and initialization. It contains the system bootstrap, system image, system directory (created as the first directory on the volume), and at least the minimum subset of system files needed to constitute a viable system.

MPX-32 refers to the system volume by the keyword SYSTEM and treats it as a public volume. All users can access the volume without issuing any further mount requests. Additionally, since the system volume is a public volume, it can be used to acquire temporary files, swap files, and spooled input/output files.

The system volume cannot be dismounted from a running system and cannot be mounted as a multiprocesor volume.

Note: The MPX-32 operating system can be configured with no system volume when using 3.0 or later revisions of Reflective Memory System Software (RMSS). See the RMSS manual for more information.

4.8.2.2 User Volume

A user volume is any volume mounted on a single processor after IPL. Unlike the system volume, user volumes are not required to have bootstraps, system images, or system files. A user volume is mounted either as public or nonpublic (see the Multiprocessor Volume section in this chapter for an explanation of these attributes).

A user volume can be physically dismounted while the system is running, provided that the volume's use and assign counts indicate the volume is not in use.

4.8.2.3 Multiprocessor Volume

A multiprocessor volume is a specially mounted user volume that allows tasks operating in separate system environments to concurrently access any volume resource. To mount a user volume as multiprocessor, the mount request must specify the SYSID option and the requested device must be hardware and software configured as a multiport device. Like user volumes, multiprocessor volumes are mounted as public or nonpublic. In order to mount a volume as multiprocessor, the following requirements must be met:

- The disk drive must be hardware configured as a dual-ported disk drive (only model 8055 and 8060 disk processors support dual-ported access). If a cache disk accelerator is used, it must be hardware-configured as multiported.
- The disk drive must be identified as a multiported drive in the appropriate SYSGEN DEVICE directive.
- The mount request must specify a SYSID parameter to identify the software port for the caller's operating environment.

4.8.3 Access Attributes for Formatted Volumes

Access to a formatted volume is determined by an attribute assigned when the volume is physically mounted to a device. In the mount request, the requesting task or user specifies either the public or nonpublic attribute. This attribute determines how all subsequent tasks and users mount or dismount the volume.

4.8.3.1 Public Attribute

A volume mounted as public is available for resource assignments by all tasks subsequently activated on the system. No logical mount is required for the tasks to gain access to the resources on a public volume.

Public volumes can store temporary files, swap files, and spooled input/output files. The system volume and swap volume (if different from the system volume) are automatically mounted as public volumes by the system initialization process.

Physical mount or dismount of a public volume can only be requested by the system administrator. If there are current users on the volume, dismount is postponed and completes only when the volume resources' use and assign counts indicate that the volume is not in use.

4.8.3.2 Nonpublic Attribute

A nonpublic volume is a volume assigned specifically to the tasks that mount it. Any task or user can request a physical mount or dismount of a nonpublic volume. Once the volume is physically mounted on a device, each task needing access to its resources must request a logical mount of the volume. The Resource Management Module maintains use and resource assign counts on the volume for system accounting.

4.8.4 Mounting Formatted Volumes

MPX-32 distinguishes two types of mounts: physical mount and logical mount. A physical mount attaches a volume to a specific device and designates the volume as public or nonpublic. A logical mount attaches a particular task or TSM environment to a physically mounted nonpublic volume.

4.8.4.1 Physical Mount

Each volume must be physically mounted to a device before the volume's resources can be assigned or allocated. When a task or user issues a mount request for a volume, the mount is completed by J.MOUNT interacting with the system operator.

To request the physical mount of a volume, users or tasks issue the TSM $MOUNT directive or OPCOM MOUNT directive, call the M.MOUNT system service (M_MOUNT in base mode), or supply the proper device RRS entry when activating a task with the M.PTSK (M_PTSK) service. The mount request specifies the device where the volume should be mounted, whether the volume is public, and, optionally, if the volume is multiprocessor. Only the system administrator can specify a volume as public.

Each mount request activates J.MOUNT, the non-resident media mount task, which issues the following mount instruction to the system console:

```
MOUNT VOLUME volname ON devmnc
REPLY R, H, A, OR DEVICE.
```

volname is the 1- to 16-character blank-filled, left-justified name given to the volume when it was formatted by the Volume Formatter.

devmnc is the device mnemonic for the unit where the volume will be mounted. If a specific channel and subaddress were specified in the mount request, a specific drive is selected and named in the message. Otherwise, the system selects a unit and names its complete address in the message.

`R,H,A,DEVICE`
are the possible responses. `R` allocates the device listed in the message and resumes the task. `H` holds the task with the device deallocated. `A` aborts the task. `DEVICE` allows the user to specify a different device to allocate.

The system operator informs J.MOUNT when the volume is installed on the requested device. J.MOUNT then notifies the task or user who requested the physical mount.

During physical mount, mount messages that require operator response display at the system console. To inhibit the mount messages and operator intervention, specify the NOMSG option in the mount request, the SYSGEN SNOP option at system initialization, or the OPCOM MODE SNOP directive.

Once a volume is physically mounted, J.MOUNT reads the volume descriptor, initializes a memory-resident mounted volume table (MVT) entry for the volume, and verifies the volume integrity.

4.8.4.2 Logical Mount

In addition to a physical mount, all nonpublic volumes also require a logical mount before a task can access their resources. A logical mount causes allocation of a volume assignment table (VAT) entry in the task's task service area (TSA). The VAT entry is linked to the volume's MVT entry, provided the volume is physically mounted. The volume's use count is incremented in the MVT entry. The task becomes a user of the volume for subsequent resource assignments and system accounting.

J.TSM handles the logical mount request in a different manner. The first logical mount request for a nonpublic volume by a TSM user mounts the task J.TSM to the volume. After the first logical mount completes, subsequent logical mounts by TSM users do not affect the use count in the VAT or MVT.

A request for logical mount is either explicit or implicit.

Explicit Mount Request — A task or user issues an explicit request for logical mount through the TSM $MOUNT directive or M.MOUNT system service (M_MOUNT in base mode). It may be accomplished as a static assignment or as a dynamic mount request during task execution.

Implicit Mount Request — An implicit request for logical mount issues as the result of TSM task activation, a log-on attempt, execution of a $JOB statement with an ownername supplied, or a physical mount request for a nonpublic volume. During task activation, all nonpublic volumes currently mounted in the user's TSM environment and the task's default working volume (if nonpublic) are implicitly logically mounted to the task. An implicit logical mount also occurs as the result of a physical mount request in any form except the OPCOM mount request.

4.8.5 Dismounting Formatted Volumes

MPX-32 also distinguishes two types of volume dismounts: logical dismounts and physical dismounts. Logical dismount detaches a particular task or user from a nonpublic volume. Physical dismount detaches a volume from the device where it is physically mounted.

4.8.5.1 Logical Dismount

When a task no longer requires the use of a nonpublic volume, it logically dismounts the volume. Logical dismount detaches a task from a volume by decrementing the volume's use count in its mounted volume table (MVT) entry and updating the task's VAT entry.

A request for logical dismount is either explicit or implicit.

Explicit Dismount Request — A task can explicitly request a logical dismount of a volume through the TSM $DISMOUNT directive or the M.DMOUNT system service (M_DISMOUNT in base mode) with no CNP options necessary. For logical dismount requests, device specificaton is optional.

Implicit Dismount Request — An implicit request for logical dismount issues as a result of task exit, end of job, or a TSM $EXIT command. When a task exits, an implicit dismount is performed for all volumes assigned in the task's VAT. A physical dismount request also causes an implicit logical dismount, unless issued using OPCOM.

A TSM logical dismount request only restricts the use of the nonpublic volume resource for the TSM user who issued the request. Any other TSM users who have the volume mounted in their TSM environment still have access to the volume resources. A task logical dismount of the volume from J.TSM occurs only as a result of a logical dismount by the last TSM user environment to have the volume mounted. Public volumes do not require logical dismount.

4.8.5.2 Physical Dismount

To remove a volume from a device, a task or user must explicitly request a physical dismount of the volume. Physical dismount is requested through the TSM $DISMOUNT directive, OPCOM DISMOUNT directive, or the M.DMOUNT service (M_DISMOUNT in base mode) using a CNP option with bit 0 of the option word set. The dismount request specifies the volume name, the device where the volume is mounted, and whether the volume is public.

Each dismount request activates J.MOUNT to handle the dismount. If there are no other users or tasks on the volume (the MVT entry use count is 0), J.MOUNT initiates a physical dismount by updating the volume descriptor, deallocating the mount device, and clearing the MVT entry. J.MOUNT then issues a message to the system console to inform the operator that the dismount has completed. It also signals by owner name the user who last requested the volume's physical dismount. If operator intervention is applicable, the operator must confirm the dismount of the volume. For removable media, this insures the operator understands when it is safe to remove the disk pack from the drive.

If there are current users on the volume (the MVT entry use count is greater than 0) when a physical dismount is requested, the volume is placed in a state of pending dismount. All subsequent requests to mount the nonpublic volume are denied. Public volume resource access is still allowed. When the last user or task detaches from the volume, or the last resource is deallocated on the volume, physical dismount completes as described.

A physical dismount can be performed for a volume recognized by MPX-32 — even if the volume is not present or the device containing the volume is not online. This condition occurs when a volume was improperly removed, the drive was improperly shut down, or the drive malfunctioned. A physical dismount clears the MVT entry from the system. It also protects any volume that may have been placed in a drive where a volume was incorrectly removed. If the volume names do not match, MPX-32 does not update the information about the volume which is presently being dismounted.

During physical dismount, dismount messages requiring operator response display at the system console. To inhibit the dismount messages and operator intervention, specify the NOMSG option in the dismount request, the SYSGEN SNOP option at system initialization, or the OPCOM MODE SIMM or SNOP directive at any time.

Only the system administrator can request the physical dismount of public volumes.

4.8.6 Automatic Mounting at System Boot

The task SYSINIT automatically mounts the system volume as part of the IPL process. The volume mounted by SYSINIT is always given the system and public volume attributes.

SYSINIT mounts the swap volume, if requested by the operator in response to a system prompt. After the swap volume is mounted, it is used by the system swapper (J.SWAPR) for swap file space allocations. A new swap volume may be established whenever the system is booted. There is no requirement to generate a new system image via SYSGEN just to change the swap volume.

If a specific swap volume is not requested at IPL, the system volume is used as the swap volume. The swap volume is always given the public volume attribute.

4.8.7 Components of a Volume

A volume is comprised of several components that are used for particular functions. How a volume's structure is defined and how it is used determines the volume's total functionality.

The component structures are:

- boot block
- volume resource descriptor
- volume root directory descriptor
- resource descriptors
- resource descriptor allocation map
- volume root directory
- space allocation map

The boot block and volume descriptor must reside in absolute fixed locations; other portions are located in either relative fixed or nonfixed locations. See Figure 4-1 for a description of the volume format.

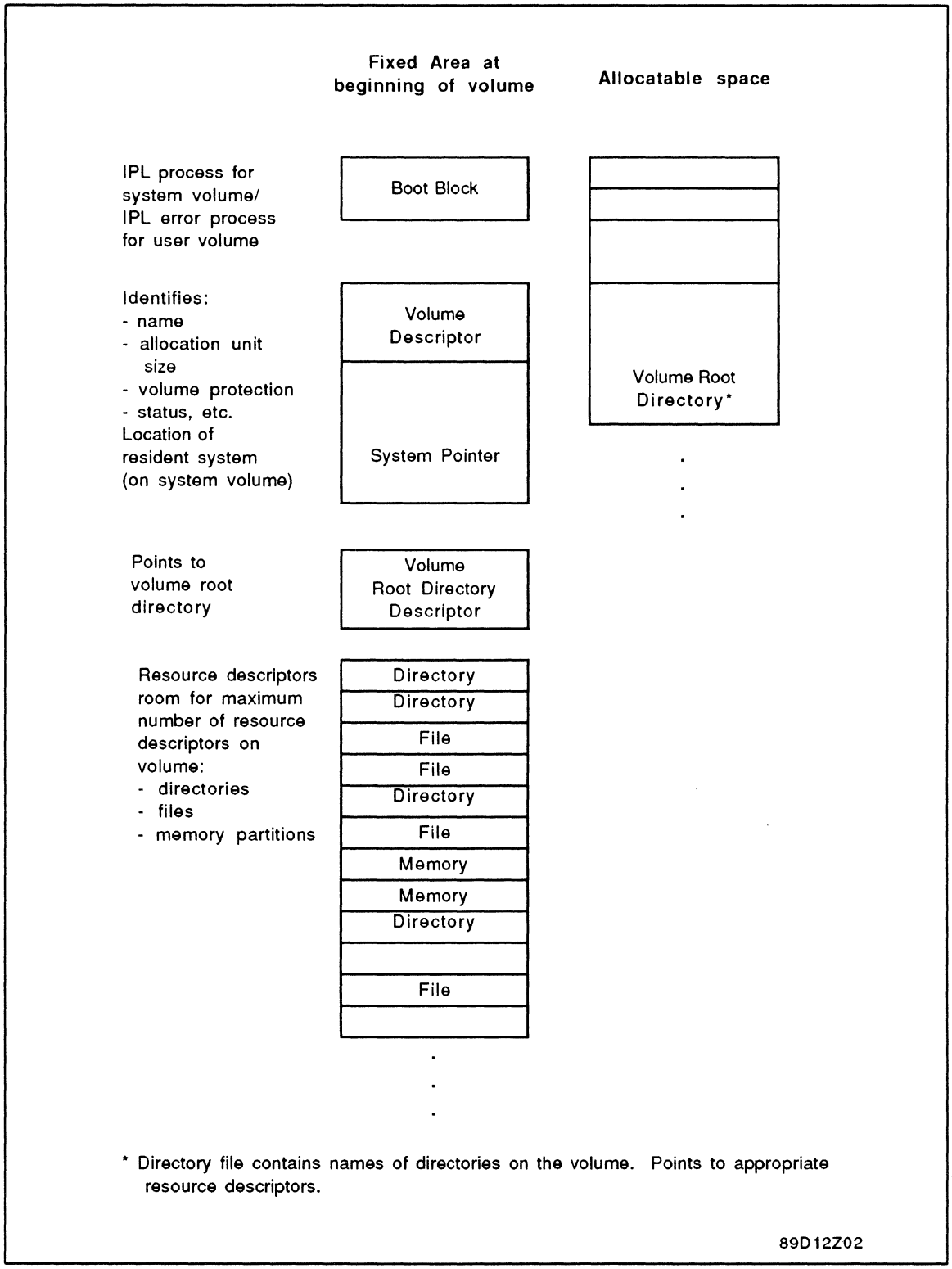


Figure 4-1
Volume Format

4.8.7.1 Boot Block

The system volume is mounted and used for IPL. When this is done, the boot block is read into the system memory and executed, bringing in the resident system image. The process is called a bootstrap. The boot block consists of a fixed number of the first consecutive blocks on the volume. They are located at the beginning of the volume to simplify the IPL process. For standard disk devices, the boot block begins at head 0, track 0, sector 0.

The number of blocks dedicated to the boot block is determined as some common denominator between the currently used sectoring (192W) and sectoring which is power-of-two related. The boot block size will be large enough to facilitate the bootstrap process currently required.

A bootstrap always occupies the boot block. The difference between system versus user volumes is that system volumes have a system image.

4.8.7.2 Volume Descriptor

The volume descriptor is a form of resource descriptor used to define the volume on which it resides. Its contents define protection and access privileges of the volume. It identifies the name of the volume, whether it can be mounted as a system volume, the volume's granularity, and other attributes, all of which are specified when the volume is created. The volume descriptor is in a fixed location and is the next consecutive block following the boot block.

The volume descriptor also points to key structures on the volume by giving their location (starting block). The volume descriptor identifies the location of the resource descriptor segments. Two segments are allowed, the first of which is obtained when the volume is created. A segment descriptor is contained within the volume which points to this segment and describes the amount of space allocated for descriptors (number of descriptors/blocks). One additional entry is provided to enable obtaining additional space for descriptors.

The volume descriptor contains a bootstrap descriptor which contains information required by the bootstrap about the system image and the device on which it resides. The bootstrap descriptor also describes the amount of space required for the system image. Entry space is also provided in the bootstrap descriptor to hold a pointer to the descriptor of a file having an alternate system image. Either image can be used at system IPL.

Other information required by the mounting/dismounting process is kept in the volume descriptor. Some of this information is also brought into an in-memory table when the volume is attached (mounted). The volume descriptor is structured so the information required for in-memory use (once the volume has been mounted) can be moved into memory simply.

4.8.7.3 Resource Descriptors (RDs)

The resource descriptor describes a particular resource. Its contents define attributes of the resource, its protection, requirements, limitations, etc.

Different types of descriptors are used to describe the generic resources made available and managed by the operating system. The different descriptor tables are referred to as directory resource descriptors, file resource descriptors, volume resource descriptors, and memory resource descriptors.

The different types of resource descriptors are intermixed within the space allocated for resource descriptors and identified by type by a fixed position field within the resource descriptor. The initial amount of space reserved for obtaining resource descriptors is specified when a volume is created. The creator also specifies whether the space obtained for resource descriptors can be increased.

Resource descriptors are allocated and deallocated from the resource descriptor segment and tracked via a bit map. Each segment contains its own bit map located at the end of the segment. When the first segment no longer contains any free descriptors, a second segment is obtained, if allowed. The descriptor is then obtained from a second segment. All subsequent requests are then obtained from the second segment until resource descriptors from the first become available.

All resource descriptors have three sections: a section containing information common to all types of resource descriptors, a section containing information different for each type, and a section for user supplied information. Certain resource descriptor information can only be changed by the system.

Copies of permanent file resource descriptors can reside in the memory resident descriptor table (MDT). This eliminates the disk access necessary to read the RDs, and results in a reduction in the amount of time spent when allocating files. Refer to the Rapid File Allocation Utility (J.MDTI) chapter in MPX-32 Reference Volume II for details.

4.9 Directories

A directory is a list of names of resources, where the entry for each name points to a resource descriptor (RD) that defines the basic characteristics of the resource (protection, starting/ending sectors, etc.). The MPX-32 operating system directory management is based on a two-level structure. Directories are:

- created on a volume
- named and protected when created as specified by the user
- associated with a user at logon (each user has a current working directory associated with the logon owner name)
- changeable; for example, users can change their current working directory

Figure 4-2 illustrates the two-level directory structure. It deals with a disk volume containing directories and files.

In the figure, alphabetic characters are used to represent directory names, numbers are used to represent file names.

The following basic concepts are related to the operating system and the two-level directory structure:

- volume root directory
- user directories
- access by pathnames
- protection

Conceptually, user directories are all the directories originating to the right of the volume root in Figure 4-2. All directory access begins at a volume root directory. The user can move to a different directory or file.

Protection is the means of restricting a user's access to a directory or file. For example, if directory Y is protected, the user may or may not be able to access file 40 and if file 40 is protected, the user may or may not be able to modify file 40 or perform other types of operations on the file.

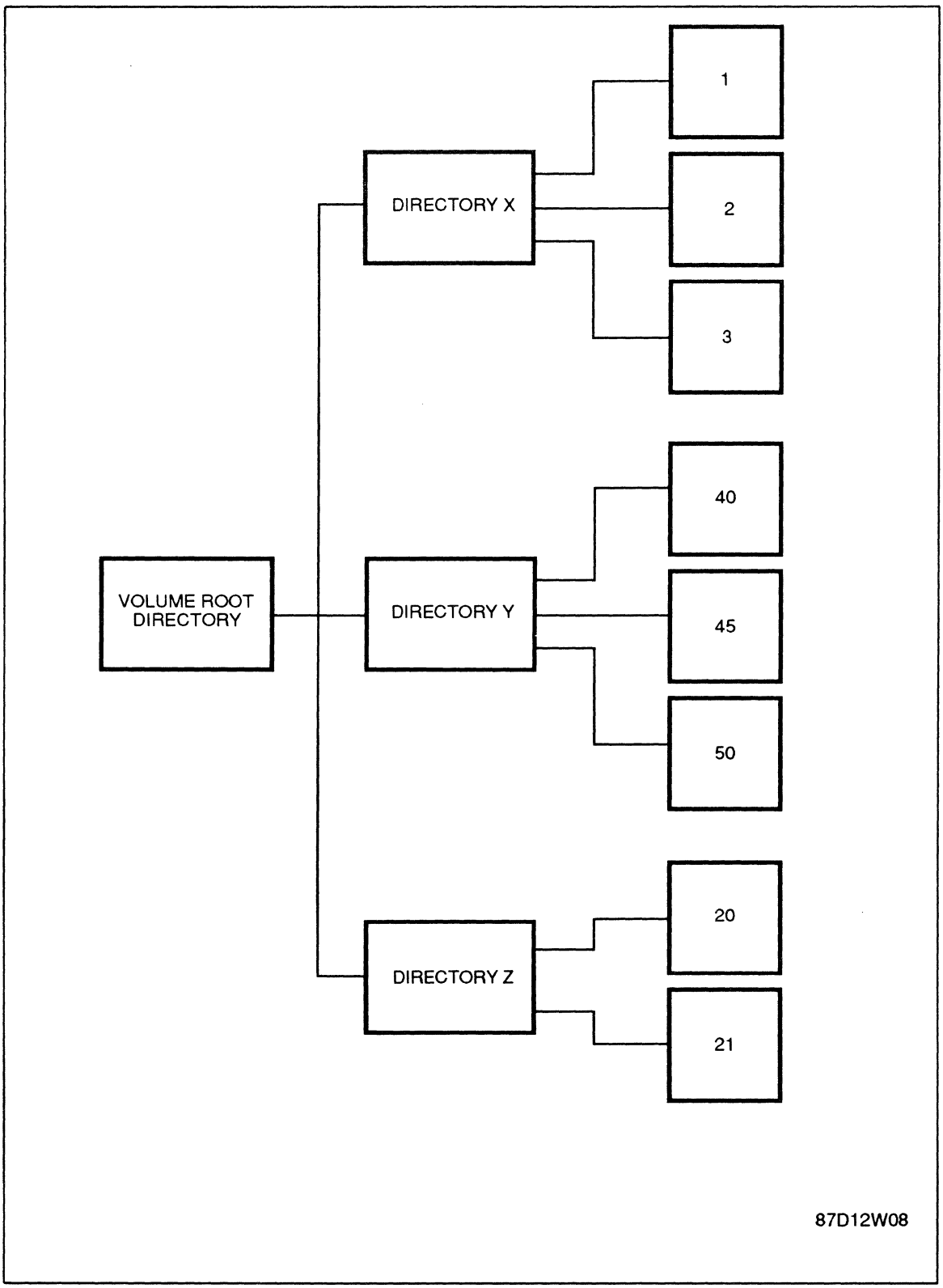


Figure 4-2
A Sample Hierarchical Directory Structure

4.9.1 Volume Root Directory

MPX-32 provides the ability to put all files on a system volume and also allows users to dismount disk packs (as user volumes). This is the concept of a root directory for a volume.

A volume root directory is maintained on every volume (see Figure 4-1) and lists the names of the directories and resources (normally files) on the volume. To get from one directory to another, the user starts from the volume root directory by using the special character uparrow (^) to go to the current volume root directory.

4.9.2 Creating Directories

The create directory function creates a directory and defines its protection and other attributes. Figure 4-3 illustrates the directory function where a user of directory X wants to access file 3. (Also see Figure 4-2).

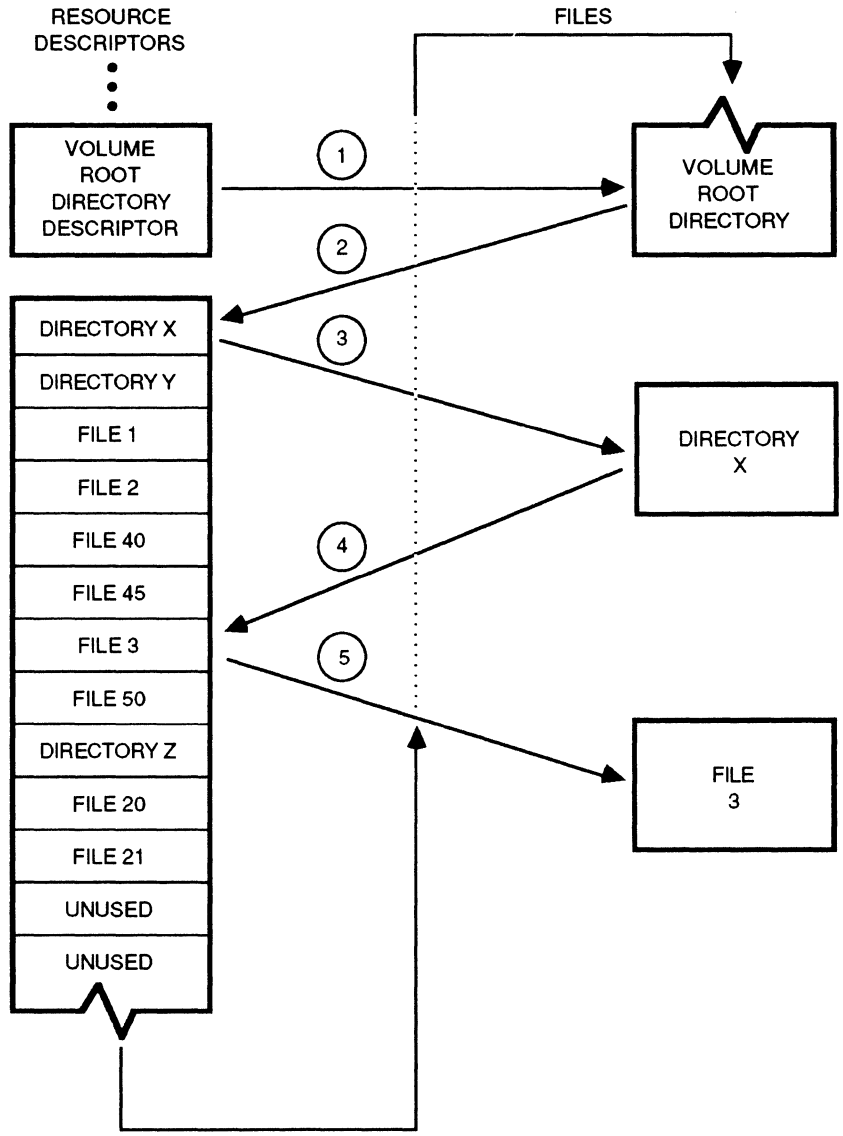
@volume^(directory)file
^(directory)file

The user must have the ability to traverse the volume root directory and to add entries in the volume root directory. The user gains access to the volume root directory as its owner, as a member of its defined project group, or as other.

To create a directory, the owner provides information which is stored in the resource descriptor for the directory:

- owner name — the name of the owner of the directory; the name can be different from the owner's logon name.
- project group name — the name of a group of users identified by the owner to have specific access privileges to the directory
- protection — the set of operations allowed separately for the owner, the defined project group name, and all others

Once a directory has been created, entries for files or memory partitions are defined using the create function.



NOTES:

1. THE ROOT DIRECTORY DESCRIPTOR POINTS TO THE VOLUME ROOT DIRECTORY.
2. THE VOLUME ROOT DIRECTORY CONTAINS THE NAME OF DIRECTORY X AND POINTS TO ITS RESOURCE DESCRIPTOR.
3. THE RESOURCE DESCRIPTOR FOR X POINTS TO DIRECTORY X.
4. DIRECTORY X CONTAINS THE NAME OF FILE 3 AND POINTS TO THE RESOURCE DESCRIPTOR FOR THE FILE.
5. THE RESOURCE DESCRIPTOR FOR FILE 3 POINTS TO FILE 3.

87D12W09

Figure 4-3
Locating a File on a Volume

4.9.3 Protecting Directories

The creator of a directory can allow or restrict the ability to read the resource descriptor for the directory and to delete the directory.

- read — allows the directory to be read
- delete — allows the directory to be deleted
- traverse — allows the directory to be traversed via a pathname

These access rights can be applied separately to the owner, the project group, and all other users.

4.9.4 Protecting Directory Entries

The creator of a directory can allow or restrict the ability to add resources to or delete resources from the directory he creates.

- add — allows additions to the directory
- delete — allows deletions from the directory

These access rights can be applied separately to the owner, the project group, and all other users.

4.9.5 Using Directories

Since all files and memory partitions are located in directories, different directories are traversed in the process of creating and manipulating resources. Although the ability to associate a working directory with a user at logon relieves the naive user from having to know about directory access, many users will be accessing more than one directory.

The SA establishes a special system file and other files that associate a working directory with a user at logon. All files the user creates are automatically located in this directory unless the user specifies otherwise. To access a resource within the current working directory, only the name of the desired resource needs to be supplied.

The working directory associated with an owner name can be changed to a different directory by using the change directory function or service and supplying the pathname to the new working directory.

What the user can do in the new working directory depends on the protection defined for the directory and whether the user owns it, supplies a project group name to gain project group access rights to it, or is another user (other).

To locate a file in a directory other than the current working directory, a directory pathname ending with the name of the file must be specified. To locate a directory other than the current working directory, a pathname ending with the name of the directory must be specified. Refer to the Pathnames section in this chapter for a description of pathname conventions.

4.10 Files

Files are sets of information stored on a volume. A file is given a unique identity so it can be referenced as a single entity for processing.

Files can store data, transactions, executable code, command sequences, etc. This section considers files as resources and does not deal with the structure of information within files.

Two types of files are allowed by the operating system: permanent and temporary. Permanent files remain defined to the operating system until they are explicitly deleted and they can be referred to in two ways: by their given name or by a unique identifier (RID) assigned by the system which allows faster access to information about the file. Temporary files are defined in the operating system as long as the task requiring them is in execution and then they are automatically deleted.

Files are attached for use either statically or dynamically. Attachment for files is a two phase process, where the first phase is assignment. Once assigned, they can be opened for use in a particular access mode. The requestor then operates on the file according to the allowed access. Files can be extended dynamically to obtain additional space. Shareable files can be accessed by multiple users. Sharing is normally restricted to compatible access modes.

A fast access mechanism is supported by MPX-32 which enables referencing resources (directories, files, memory partitions) by their resource identifier (RID). All resources defined on a volume have an associated RID. The creator of the resource can obtain the resource's RID by specifying the appropriate address in the resource create block (RCB) when the resource is created. See Chapter 5, Tables 5-16, 5-17 and 5-18. This fast access mechanism should not be confused with the fast access attribute which only applies to files. Refer to the section, Fast Access Attribute in Chapter 3 of this volume.

4.10.1 File Attributes

The basic attributes of a file are defined when it is created and include:

- size
- extendibility
- access privileges
- sharability
- contiguity
- fast access
- protection

These attributes can only be altered while the file remains defined to the system.

4.10.2 Obtaining File Space

Space is obtained for a file out of unused space on a volume. The user requests space initially by the create function and subsequently by an extend function. A file can be extended automatically by setting up appropriate parameters when the file is created.

The space obtained for a file is marked used to prevent duplicate use of the same space. The space obtained for a file in any single allocation request is called a segment. Files containing a single segment are contiguous, while those containing multiple segments may not be contiguous.

There are certain efficiency tradeoffs resulting from obtaining space in different ways. The most efficient use of space is gained from obtaining contiguous space. This can be ensured by requesting the total amount of required space when the file is created and by declaring the file nonextendible. Efficiency gained by using contiguous space is most noticeable when randomly accessing a file. Files that need to be extendible can gain some efficiency by extending space in fixed length segments. Although this is not as efficient as creating a single contiguous file space, it is a significant improvement over extending in variable length segments.

4.10.2.1 Granularity

The space on a disk volume is measured in units known as blocks. A block is the logical sector size of all disk volumes the operating system supports. The block size used is 192 words. To minimize fragmentation of unused space resulting in inefficient usage of the available space, the space is obtained in allocation units consisting of a fixed number of blocks.

The allocation unit size is determined only at the creation of a volume and can be any number of blocks. The default size for an allocation unit is determined by the class of the device on which the volume is mounted at the time it is created. Once an allocation unit size is established for a given volume, it is fixed and does not change. Although the volume can be subsequently mounted on other devices of the same class, its allocation unit size remains unchanged.

4.10.2.2 Contiguity

For efficiency, it is desirable to obtain contiguous space for a file. In a contiguous file, space is comprised of a single segment. A file is defined as contiguous when it is created by indicating it is not extendible, and by not specifying multiple segment creation option.

4.10.2.3 Extendibility

Frequently, it is difficult to determine all the space a file requires at creation and it is necessary to be able to extend the file's space according to changing needs. If specified at creation, a file's space can be extended beyond its current size.

For extendible files, it is desirable to specify the lengths (number of blocks) in which a file will grow. The length is specified when the file is created and becomes the segment size to use whenever the file is extended. The segment size specified is rounded to the next highest allocation unit defined for the volume on which the file is created. If a segment size is not specified, a system default is applied. If the file is created with the zero option specified, any extensions to the file are first zeroed.

A file can be extended by either fixed or variable length segments. A fixed length segment has its maximum increment equal to its minimum increment. A variable length segment has its maximum increment greater than its minimum increment. For fixed length segments, if the contiguous space required is not available, the file is not extended. For variable length segments, the requested amount of space is obtained except where the request is greater than any available contiguous space. In this case, the largest available amount of contiguous space is obtained and the user is notified.

A file can be extended either manually or automatically. This is also defined at creation of the file. For automatic extensions, the user of the file need not be aware of requirements for additional space. When required, the file is extended without notification. For manual extensions, the user is notified when more space is needed. Optionally, the user extends the file via the extend function. If variable length extensions are requested, the requested size extension is attempted. If the requested size cannot be obtained, a second attempt is made for the maximum increment size. If the maximum increment size cannot be obtained, an attempt is made for minimum increment size. If none of the three sizes can be obtained, a denial is issued. For both automatic and manual extensions requiring fixed size segments, if the request exceeds the available contiguous space, the request is denied.

4.10.2.4 Size

A file's size is the amount of space obtained for it. The size of a file is determined at creation of the file or by extending the file.

The size specified when a file is created is the minimum space allocated to the file (initial space allocation). The close function will cause a truncation of the file's space to the file's initial space or its minimum space requirement.

The maximum size allowed for a particular file to be extended can be specified at creation of the file. The maximum size for the file can also be specified to prevent users from extending a file beyond prescribed limits.

4.10.3 File Names and Fast Access

It is desirable to reference files in a simple fashion. Files are given symbolic names at their creation and are then referenced by their name. Such files are called permanent files. Files created without specifying a name are called temporary files.

All temporary and permanent files are known internally by a unique file identifier (RID). This identifier is assigned internally at creation of the file and identifies the resource descriptor block which defines the respective file's attributes and space.

When referencing a file by its name, a given amount of time (overhead) is required to obtain its definition from a directory. This can be excessive for some time-critical applications. To eliminate the overhead of directory searches, a file may be referenced by its RID. All temporary and permanent files can be referenced by their RID. When a file is created and the fast access attribute is specified, the file's RID remains unchanged throughout any operation performed on the file. For further details, refer to Fast Access Attribute section in Chapter 3.

4.10.4 File Protection

In many applications, it is necessary to protect files from certain users and types of access. The users of a file are granted its use only in the ways allowed them. Protection can be applied to safeguard attaching and/or extending files.

Protection of files is established by the creator/owner at creation of the file. Only the owner of the file can specify or modify the file's protection attributes.

4.10.5 Permanent Files

Permanent files are named files and are permanent to the system until explicitly deleted. They are known to the system by one directory entry and are defined to the system at their creation. There can be one and only one definition for a named file. This definition is maintained and known to the system via a resource descriptor.

Permanent files can be used either shared or nonshared. This is determined by the owner/creator. Permanent files are shared according to rules defined previously.

The allowed uses of a file can be altered at creation, assignment, and open. For files, attachment is comprised of the last two phases (assign and open). By the time a task accesses a file, all uses have been defined and verified. How each of these functions alters the context of file use is described in the following sections.

4.10.6 Creating Files

The create function allocates file space, defines the attributes of a file, and builds a resource descriptor which defines the file space. This resource descriptor also contains the attributes of the file.

For permanent files, the user-supplied file name associates the directory entry, resource descriptor (file attributes), and the file space. For temporary files and fast access files, a unique file identifier is supplied by the system. This provides an alternate method of association with a file and its attributes.

Files exist when they are created by the operating system. The creation of a file is accomplished as requested by the successful completion of an executed create function. Once a file is created, it can be attached and accessed by its name or identifier.

At creation, the attributes of a file are defined either explicitly, by providing them as specified parameters or implicitly, by omitting certain parameters. If parameters are omitted, reasonable defaults, values, or attributes are assumed. If there is no reasonable default for a parameter, the parameter must be supplied by the user.

4.10.7 Attaching Files

To secure a file for use, the file must be attached. When attachment of a file is requested, the requestor is granted attachment on the basis of both the file's defined allowances to the requestor and its availability. The two phases required for the attachment of files are assignment and open.

4.10.8 Assigning Files

File assigning by the assign function is an attachment phase that associates a resource with a task by a logical file code.

Nonshareable files allow only exclusive use. Once nonshareable files are assigned, they cannot be used by others until released. A requestor of a nonshareable file that is already assigned is enqueued or optionally denied on the request.

A requestor can also gain exclusive use of a shareable file by requesting its exclusive assignment. As in the previous case, the requestor is enqueued or optionally denied if the file is currently assigned to another task.

Shareable files can be shared in two ways: implicitly or explicitly. If shared use is requested when the file is assigned, the file is explicitly shared. Assignment to a file for explicit sharing is allowed only if there are no other tasks attached to the file or if all others who are attached are explicitly sharing the file.

If neither shared or exclusive is specified when assigning a shareable file, it is implicitly shared. This is allowed only if no other tasks are attached to the file or if all other tasks are implicitly sharing the file in a compatible access mode.

In summary, the first task that attaches a file establishes the context of use for subsequent requestors of the file. The context established by the first task can be changed only when the file is detached from the tasks.

The desired access mode(s) for the requested file may be specified when assigning the file. This defines the intended access for the file attachment.

A specific access mode or multiple access modes can be requested when assigning the file for implicitly shared use. The attachment is granted if the requested access mode(s) are compatible with those of users currently attached to the file.

Access modes can be omitted when assigning a file for implicit shared use. In this case, the only access ensured is read access. Other access modes can be requested when later opening the file. As a result, the requestor may be enqueued for the request since there is no guarantee the requested access mode will be compatible with other users.

The required access mode(s) need not be specified for explicitly shared files. For explicit sharing, there is no compatibility requirement since all sharers are expected to synchronize and use locking to maintain file integrity. It is not possible for explicit sharers to contend for file use as a result of specified access modes when opening the file. They are guaranteed use of the file in the requested mode.

Assignment parameters are defined explicitly by providing them as specified parameters or implicitly by default.

4.10.9 Opening Files

A file must be assigned before being opened. A file must be opened before any operations to the file are allowed. The desired file is referenced as a required parameter to the open function.

The access mode for a file is determined at open. The type of access allowed depends on the allowances or restrictions associated with a file at creation and assignment.

The access mode in which the file is opened determines the position within the file. The open function performs logical connections of control table information between the file, system, and its requestor. Also, if required, a device handler is initialized for the device where the file resides.

4.10.10 File Operations

The operating system provides a set of operations which can be performed on files. The data structure within the file itself is of no concern to the operating system. The lowest structure recognized by the operating system is the block.

Files can be accessed either sequentially or randomly. The intended access is specified when opening the file. Files opened as sequential are operated on in a sequential manner. Subsequent operations advance one block from the previous position in the file. Files opened as random are processed or operated on in a random manner. Each operation supplies a specific file relative block number to which the operation is performed.

The access method is determined at open by examining the random access indicator contained in the file control block (FCB). If the random access indicator is not set, the access method of the file is determined to be sequential access.

Three general types of operations are provided for use with files:

- read — transfer data from a file to memory
- write — transfer data from memory to a file
- position — move to an indicated position in a file

For read and write operations, parameters are supplied denoting: the memory address to or from which the data is to be transferred, the number of data bytes to transfer and the position in the file (implied for sequential access) at which the operation is to commence. Position can also be specified independently of read or write operations, in which case, no data transfer operations are performed during the position function.

4.10.10.1 Sequential Access

Sequential access gives the user the ability to transfer data to or from a file in a sequential manner. The user is allowed to specify a buffer address aligned on an arbitrary byte boundary and specify an arbitrary transfer count in bytes. The transfer granularity to disk files is 192 words. This means transfers to or from a file are executed in multiples of 192 words, for example, disk blocks. During output operations to a disk file, only the requested number of bytes are output to the disk file. Any bytes remaining to acquire the next highest 192 word boundary are automatically zeroed by the disk controller.

The operating system recognizes file granularity. It does not recognize the data format inside a file, therefore, the file system is not sensitive to record boundaries. A file must be read in a form that is compatible with the way the file was written.

4.10.10.2 Random Access

If the random access indicator is set in the file control block (FCB) when a file is opened, the access method of the file is determined to be random access. This means the user must specify the file relative block number (192 words) where the requested read or write operation is to begin. As with sequential access, the user is allowed to specify a buffer address that starts on an arbitrary byte boundary and an arbitrary transfer count in bytes. Also, as in sequential access, the operating system is only cognizant of the 192 word granularity of a file, therefore, data formats denoting record boundaries are not detectable.

The operating system supports extendible random access files. Using auto-extendibility on random access files can cause the file to become discontiguous; therefore, the efficiency of a program performing random access disk I/O might be impaired. To prevent this, create the file with sufficient size to allow for possible extension.

If automatic file extension is to be inhibited, the file must be created with the appropriate attributes through VOLMGR. If autoextendibility is inhibited and an attempt is made to access beyond the file, an EOM indicator is set.

It is necessary to initialize the file with a known data pattern and detect null records-areas within the file that contain the initial data pattern. As an option, a file may be sequentially initialized with a pattern of binary zeroes at creation.

4.10.11 File Positioning

File positioning provides the capability for moving within a file without transferring data to or from the file. The rules for file positioning are dependent on which access method is in effect on the file. Two types of file positioning are allowed:

- absolute — the ability to position to the beginning or end-of-file
- relative — the ability to move to a location in a file with respect to the current position in the file

4.10.11.1 Absolute File Positioning Operations

Absolute positioning allows the user to position to the beginning-of-file or to the end-of-file without regard to the current position in the file. Absolute positioning is used with sequential and random access methods. Three operations are provided for absolute positioning in a file:

- rewind file — position to the beginning-of-file and indicate beginning-of-medium
- backspace file — position to the beginning-of-file. Same as rewind.
- advance file — position to the end-of-file and indicate end-of-file. Any attempt to advance a file beyond end-of-file causes an end-of-medium to be indicated.

4.10.11.2 Relative File Positioning Operations

Relative positioning allows the user to position to the beginning-of-record or to the end-of-record with respect to the current position in the file. Relative positioning can be used only with the sequential access method. Two operations are provided for relative positioning in a file:

- backspace record — backspace a file block (192 words). The beginning-of-medium indicator is set if this condition is detected.
- advance record — advance a file block (192 words). The end-of-file indicator is set if this condition is detected. Additionally, the end-of-medium indicator is set if positioning beyond end-of-file is attempted.

4.10.12 File Access Modes

File access modes control allowed access methods and combinations of operations allowed to a file. The operating system defines five allowable access modes: read, write, modify, update and append.

The mode in which a file is to be accessed is specified when the file is opened. Only one access mode can be specified at open.

The following chart shows the access modes and the conditions determined when the file is opened.

Table 4-1
File Access Modes and Conditions

Requested Mode	Allowed Operation	Allowed Access Methods	File Position at Open	EOF Position at Open	EOF Position at Close
Read	Read, ABS Position REL Position	Sequential Random	First block of file	Highest sequentially written block number + 1	Same position as at open
Write	Read, Write, ABS Position, REL Position	Sequential only	First block of file	First block of file	Highest sequentially written block number + 1
Modify	Read, Write ABS Position, REL Positon	Sequential, Random	First block of file	Highest sequentially written block number + 1	Same position as at open
Update	Read, Write, ABS Position, REL Position	Sequential, Random	First block of file	Highest sequentially written block number + 1	Equal to position at open or new highest sequentially written block number + 1 if data was appended to the file
Append	Read, Write, ABS Position, Rel Position	Sequential only	* Highest sequentially written block number	Highest sequentially written block number + 1	New highest sequentially written block number + 1

- EOF refers to the actual end-of-file block (or defined end-of-file), not to any software end-of-file (either blocked or unblocked) contained within the file.
- The EOF block is equal to the first block of a file when the file is created.
- The EOF block is equal to the last block of a file if the file is optionally initialized with the binary zero pattern when the file is created.
- * For blocked files, the file position at open is placed at the last blocked software end-of-file before the defined end-of-file block.

4.10.12.1 Read Mode

A file opened in read mode allows read-only access to a file. The position of the file after opening is the first block of the file. Read operations operate from the first block of the file to the defined end-of-file.

Sequential and random access methods are allowed in the read mode. Additionally, absolute and relative positioning is allowed with respect to the restrictions appropriate to blocked files.

4.10.12.2 Write Mode

A file opened in write mode allows sequential read and write access to a file. The position of the file after opening is the first block of the file. Write operations operate from the first block of the file to the last block of the file's allocated space (EOM). With extendible files, additional file space can be automatically allocated when the EOM condition is detected and the file is being written.

The write mode is provided to write the initial data contents of a newly created file. Write mode can also establish new data contents for a file (i.e., file rewrite). When a file is opened in write mode, the end-of-file (EOF) indicator is automatically set to the first block of the file. This step effectively discards the current data contents of the file. As the file is sequentially written, the end-of-file indicator is moved and logically exists at the end of all data that has been recorded in the file. Operating in this manner allows the file, if shared, to be read while a single writer is establishing new data contents for the file. This method of operation is only allowed when the writer is the first task to open the file. Readers opening the file after the writer are able to read all data the writer has written. Attempts by any reader to read data beyond the writer's current position in the file cause that reader to be suspended (i.e., blocked from execution), until the writer has established additional new data in the file. The results of this method of operation are only predictable when all sharers of the file are accessing it sequentially. If the writer cannot be certain of sequential access, the writer should attach (assign or open) the file for exclusive use.

When a file opened in the write mode is closed, the end-of-file position is recorded in the resource descriptor for the file. This enables determination of the required size of the file, and all existing and unused extensions to the space of the file are returned to the pool of allocatable space on the volume.

Only the sequential access method is allowed in the write mode. Additionally, absolute and relative positioning is allowed but discouraged if the file is to be concurrently shared with readers on the file.

4.10.12.3 Modify Mode

A file opened in modify mode enables read and write access to a file. The position of the file after opening is to the first block of the file. Modify operations operate from the first block of the file to the defined end-of-file.

The modify mode is provided to allow modifications to be made to the existing data contents of a file.

For blocked files, modify operations can rewrite or change the position of the blocked software end-of-file. However, modify operations are still constrained to operate within the original range of the first block of the file to the defined end-of-file block.

Sequential or random access methods are permitted in the modify mode. Since all or portions of the existing data in the file can be modified, other tasks are not able to gain concurrent access to the area of the file operated on by the modify mode. Additionally, absolute and relative positioning is allowed with respect to the restrictions appropriate to blocked files.

4.10.12.4 Update Mode

A file opened in update mode enables read and write access to a file. The position of the file after opening is to the first block of the file. Update operations operate from the first block of the file to the last block of the files allocated space (EOM). With extendible files, additional file space can be automatically allocated when the EOM condition is detected and the file is being written.

The update mode is provided to allow modifications to be made to the existing data contents of a file and to append new data to the file. Sequential and random access methods are allowed in the update mode. Since existing data in the file can be modified and new data can be appended, other tasks cannot gain concurrent access to any portion of the file.

Although sequential and random access methods are permitted, appending new data to a file must be done sequentially regardless of which access method is in effect on the file. Additionally, absolute and relative positioning is allowed with respect to the restrictions appropriate to blocked files.

4.10.12.5 Append Mode

A file opened in append mode allows new data to be appended to existing data in a file. The position of the file after opening is at the end of existing data in the file. Append operations operate from the file position at open to the last block of the file's allocated space (EOM), i.e., rewind cannot go past the file's position at open. With extendible files, additional file space can be automatically allocated when the EOM condition is detected and the file is being written.

As new data is sequentially appended to an existing file, the end-of-file indicator is moved and logically exists at the end of all data that has been appended to the file. Operating in this manner allows the file, if shared, to be read by concurrently executing tasks while new data is being appended to the file. This method of operation is only allowed when the appender is the first task to open the file. Readers opening the file after the appender are able to read all data the appender has written. Attempts by any reader to read beyond the appender's current position in the file cause that reader to be suspended (i.e., blocked from execution) until the appender has appended additional new data to the file. The results of this method of operation are only predictable when all sharers of the file are accessing it sequentially. If the appender cannot be certain of sequential access, the appender should attach (assign or open) the file for exclusive use.

When a file opened in the append mode is closed, the end-of-file is recorded in the resource descriptor for the file. Append mode, unlike write mode, does not return unused space at the end-of-file to the pool of allocatable space on the volume.

Only the sequential access method is allowed in the append mode. Additionally, absolute and relative positioning is allowed but discouraged if the file is to be concurrently shared with readers on the file.

4.10.13 Sharing Files

Shared access allows simultaneous access for users of differing access modes but places restrictions on certain combinations for implicit sharing. Users who implicitly share a file must open the file with access modes compatible with all other users of the file. The following summarizes the compatible modes for any single access mode:

Access Mode	Compatible Access Mode
Read	read, write*, or append
Write	read*
Modify	append+
Update	none
Append	read or modify+

* Read and write are compatible only if the writer is the first task attached to the file. If not, the writer must wait until the reader closes the file.

\+ Append and modify are not compatible for blocked files.

Explicit sharing of a file allows the user to intermix all access modes, some combinations of which are considered incompatible for implicit sharing. Synchronization and file locking functions can be used to ensure locking out simultaneous accesses to files when multiple writers/readers are sharing a file explicitly and could thus yield undefined results. Explicit sharers do so knowingly, and therefore, must perform their own synchronization and locking control.

4.10.14 Closing Files

Closing a file prohibits the requestor from subsequent operations to the file. The file is then closed from the perspective of the requestor. For shared files, the file does not become closed to other sharers of the file.

When a user closes a file after implicitly sharing it, the access modes available to others for the same file can change. This is determined at the close of the file. Closing can then allow the system to complete other requests for assigning or opening the file, not formerly allowed.

In some access modes, the end-of-file is determined at its close. This also allows determination of the required size of a file and enables the return of unused space. For implicitly shared files, it can also mean that areas not previously accessible to other sharers become accessible after any single-sharer closes.

4.10.15 Detaching Files

A file can be detached by requesting the deassign function. This frees the file and returns it as an available resource to the system. The freeing of the file is from the perspective of the requestor of the detachment. If the file is being shared and is attached by other sharers, the sharers maintain attachment of the file. In all cases, when a file is detached, its use count is decremented. The file is completely returned to the system when the use count is decremented to 0 (file not in use). The file then becomes available to other requestors who may have been suspended due to the file having been in use.

4.10.16 Deleting Files

Permanent files are deleted by using a delete function. When the file is deleted, the space obtained for the file is returned to the volume by marking its previously occupied allocation units as available in the volume's allocation map. The entry for the file is removed from its associated directory.

4.10.17 Temporary Files

Temporary files are a type of file resource specified and defined when they are created by the create function. They are deleted from the system when the creator of the file terminates execution.

Temporary files do not have names associated with them. They are referenced by a unique assigned identifier called a resource identifier that contains an integer index pointing directly to the file's resource descriptor.

4.10.17.1 Creating Temporary Files

Temporary files are created by executing a create function requesting a temporary file. All parameters allowed for creating permanent files are also allowed for temporary files except fast access. Because temporary files have no name and must be referenced by their assigned identifier, they are already fast access. For example, their resource descriptor can be found in one disk access. Temporary files can also be created by the assign function.

Typically, most parameters allowed for creating a temporary file are not required. When a temporary file is to be made permanent, these parameters establish the attributes of the file to use when the file becomes permanent.

4.10.17.2 Assigning Temporary Files

Temporary files must be assigned by the assign function before they can be opened. Assignment establishes the tables required to use the file and reserves the file for use by the requestor.

Existing temporary files, those having previously been created by a create function, are assigned (assign function) by using their resource identifier.

The resource identifier is given to the requestor when the file is created. The requestor also gives a logical file code (three characters) which becomes logically equated to the resource identifier. Once the file has been assigned, it can be referenced by its logical file code.

Temporary files not created by the create function can be both created and assigned by the assign function. For such cases, the initial space allocated is specified by the assigner or is a fixed number of allocation units.

Temporary files created at assignment are extendible. Reasonable defaults are assumed for parameters normally specified when creating a temporary file. The device where the temporary file is to be created can be specified indirectly by volume name.

4.10.17.3 Opening and Accessing Temporary Files

Temporary files are accessed initially by executing an open function. The access modes that the file can be opened in are specified when assigning the file. The rules for this function are the same rules that apply to permanent files.

4.10.17.4 Deleting and Detaching Temporary Files

The space used by a temporary file is freed by executing a deassign function. Deletion of a temporary file is implied when the file is detached.

Because all files are detached at termination of a task, temporary files are then implicitly deleted upon termination of the using task, whether the termination is normal or not. The task has the ability to make a temporary file permanent before it terminates.

4.10.17.5 Making Temporary Files Permanent

Temporary files can be made permanent through the use of the M.TEMPER system service (M_TEMPFILE TO PERM). Execution of this function creates an entry in the directory specified by the pathname. The entry points to the resource descriptor for the temporary file. For further information concerning directories and/or pathnames, refer to the section Directories and the section Pathnames in this chapter.

4.11 Memory Partitions – Nonbase Mode of Addressing

Memory partitions are named areas of physical memory that can be shared by concurrently executing nonbase mode tasks. Each memory partition has a relationship with physical memory and with the logical address space associated with a task. There are two types of memory partitions: static and dynamic.

Static partitions are defined when the operating system is generated using SYSGEN and are created when that system is booted. When the static partition is defined, its physical location, size, and logical location are specified. Partitions declared in this manner permanently reserve the specified physical area of memory, which remains reserved until the system is regenerated. Static memory partitions cannot be deleted.

To make the physical region available to the logical address space of a task, the task must include the partition. Certain static partitions such as GLOBAL*nn* and DPOOL*nn* have known names to the system and can be automatically included. MPX-32 specifically allows multiple definition of the same physical area. This allows the same physical area to be mapped into a different logical address space for different tasks. It also can be used to allow multiple partial map block partitions to be included in the same map block.

Dynamic memory partitions are created by system utilities. When the memory partition is defined, the user specifies the partition's relationship to the task's logical address space. The partition's relationship to physical memory, for example, its size is also specified, but the physical memory is not allocated until the dynamic partition is allocated to a task.

4.11.1 Creating Memory Partitions

When memory partitions are created, the attributes of the partitions are defined. These attributes include:

- name
- protection
- size
- location in logical address space

When a partition name is used to attach the partition, the size, location, and other attributes are validated.

4.11.2 Protecting Memory Partitions

The protection allowed to memory partitions is the same as the protection allowed to any other resource managed by the operating system, for example, owner, project group, and other. In addition to the common forms of resource protection, partitions can also be protected from write access.

4.11.3 Attaching Memory Partitions

To attach a partition, the partition must have been created. The requestor is granted access to the partition based on the partition's access rights defined for the requestor. Valid access rights for partitions are delete, read and write. For example, a particular user cannot attach a partition that is protected from the user.

Memory partitions are always attached to a task for explicit shared use. A task is not denied attachment of a shared partition if the user for whom the task is executing has the proper access rights.

Once a partition has been attached, the nonbase mode task gains access to the partition via the M.INCLUDE and M.EXCLUDE system services. The M.INCLUDE service maps the partition into the task's logical address space, providing the space for the partition is not currently allocated.

4.11.4 Accessing Memory Partitions

Once mapped into the task's logical address space, the task accesses the space of the partition by using memory reference instructions. If the user associated with the task does not have write access to the partition, the task is prevented from modifying the contents of the partition.

4.11.5 Detaching Memory Partitions

The M.EXCLUDE service allows the task to map a memory partition out of the task's logical address space. This makes the space available to include another partition or to use the space in some other manner. A partition that has been excluded from the task's address space cannot be referenced until it is again included.

Detaching memory partitions informs the system that the task no longer requires a guarantee that the partition will remain available for access. If the partition is mapped into the task's logical address space at the time of the detachment request, the partition is excluded from the task's address space.

Detachment of a static memory partition does not release the physical memory assigned to the partition nor does it modify the contents of the partition.

Detachment of a dynamic memory partition releases the physical memory assigned to the partition if no other tasks currently have the partition attached. When the physical memory used by a dynamic partition is released, the contents of the memory locations are made available for any type of physical memory request.

4.11.6 Deleting Memory Partitions

Static memory partitions cannot be deleted. Dynamic memory partitions can be deleted using the M.DELETE service.

4.11.7 Sharing Memory Partitions

Memory partitions can be attached and accessed by concurrently executing nonbase mode tasks. The users of shared partitions have the use of shared resource synchronization features.

Additionally, these users can develop their own protocol for sharing the resources.

4.12 Shared Images

Shared images are named areas of physical memory that can be shared by concurrently executing base mode tasks. Shared images can be absolute or position independent.

Absolute shared images have fixed logical addresses within a task's logical address space.

Position independent shared images do not contain any relocatable address references. Any references outside of the shared image are relative to a base used at execution time. The logical address of a position independent shared image is defined to the task at link time.

A shared image can contain both read-only and read/write program image sections. At link time, a specification can be made to activate the shared image as single copy or multicopy. A single-copy shared image (the default) has only one copy of both the read only and the read/write sections in memory. Both sections are shared by all tasks requesting inclusion of that image.

A multicopy shared image has a separate copy of the shared image for each including task. A multicopy-shared shared image has a single copy of the read only section for all tasks and a separate read/write section for each task.

4.12.1 Created Shared Images

Shared images are created by the LINKER/X32 when the attributes of the shared images are defined. These attributes include:

- name
- protection
- size
- location in logical address space

The size, location, and other attributes are validated when a shared image name attaches the shared image.

4.12.2 Protecting Shared Images

The protection allowed to shared images is defined at link time and is the same as the protection allowed to any other resource managed by the operating system. This protection is the owner, project group, and other scheme. In addition to the common forms of resource protection, shared images can also be protected from write access.

4.12.3 Attaching Shared Images

To attach a shared image, the shared image must have been created. The requestor is granted access to the shared image on the basis of the shared image's access rights defined for the requestor. Valid access rights for shared images are read and write.

The access mode is requested at link time and a denial is made at include time if the requested mode is incompatible with the shared image definition.

Shared images can be included into a task's address space by preassignment or dynamic inclusion. A preassigned image is loaded and/or mapped into the referencing task's logical address space at activation time and remains there until completion or until the task excludes it via the M_EXCLUDE or M.EXCLUDE system service. A dynamic image is loaded and/or mapped upon request by the M_INCLUDE or M.INCLUDE system service.

4.12.4 Accessing Shared Images

All shared images to be included by a task, whether preassigned or dynamic, must be defined by the user at link time.

When a shared image is linked, a version number and compatibility level can be specified. This information is copied into the preamble for each task referencing the shared image at link time, and is used to verify that the shared image requested at activation is compatible with the shared image defined at link time.

The physical address of a shared image can be specified at link time enabling systems with shared memory to share the image. This feature should be used with caution because a task is given an immediate denial if the requested physical memory is allocated to another task.

A shared image can be defined as resident at link time. A resident shared image is then loaded into memory using the OPCOM INCLUDE directive, and remains in memory until removed by the OPCOM EXCLUDE directive.

4.12.5 Detaching Shared Images

The M_EXCLUDE and M.EXCLUDE services allow the task to remove a shared image from the task's logical address space. This function makes the space available to include another shared image or to use the space in some other manner. A shared image that has been excluded from the task's address space cannot be referenced until it is again included.

If write back is requested, the read/write section is written back to the disk. If write back is not requested, there is no write back. Write back is not performed until all users have detached the shared image. The physical memory occupied by the shared image is then available to other tasks, providing the image is not included as resident.

4.13 Multiprocessor Shared Volumes

The multiprocessor shared volume is an MPX-32 feature that allows tasks, operating in separate system environments, to obtain concurrent directory and file access. The operating system maintains resource integrity against incompatible access or usage modes on these resources within the scope of volume management described in this chapter.

A volume is treated as multiprocessor only if it has been software mounted as multiprocessor on a multiported drive. A multiported drive is defined to be any disk drive that is hardware configured with the ability to communicate concurrently with up to sixteen independent processors. The hardware characteristics of the disk drive are defined by the appropriate DEVICE directive supplied to the SYSGEN utility. The software characteristics of the volume are defined by the presence or absence of a SYSID parameter when the volume is mounted.

The synchronization mechanism for multiprocessor resources is maintained by software information kept in the resource descriptor (RD). Therefore, consideration must be given to system performance and access restrictions on these resources. The specific performance and restriction issues applying to multiprocessor resources are discussed in the Multiprocessor Resource Access and Multiprocessor Resource Restrictions sections.

Note: The system volume cannot be a multiprocessor volume.

4.13.1 Multiprocessor Resources

A multiprocessor resource is defined as a volume resource residing on a volume mounted as a multiprocessor. Permanent files, temporary files, directories, the volume descriptor map (DMAP) and the volume space map (SMAP) can be multiprocessor resources. Memory partitions and space definitions are never treated as multiprocessor resources regardless of where they reside.

Because the resource synchronization mechanism for multiprocessor files must be kept on disk (in the RD) rather than in memory, additional system overhead is incurred in the areas of create, delete, assign, open, close and deassign resource on multiprocessor volumes. After a file is allocated, the actual number of I/O operations performed is not affected by the multiprocessor characteristics of the resource.

4.13.2 Multiprocessor Resource Access

When the allocation status of a multiprocessor resource changes, MPX-32 synchronizes the update of the resource accounting information for that resource using the multiprocessor lock in the last word of its resource descriptor (RD).

The multiprocessor lock acts as a semaphore for the resource. It is reserved by MPX-32 for this purpose and should not be used for any other applications.

If the multiprocessor lock is not set when MPX-32 attempts to allocate a resource, MPX-32:

- sets the lock
- updates the allocation and access information in the RD
- releases the lock

If the multiprocessor lock is set when MPX-32 attempts to allocate a resource, MPX-32:

- suspends the task for a specified length of time and then tries to allocate the resource a second time

  Use the SYSGEN DPTIMO directive to specify the length of time to suspend between tries. When DPTIMO is not specified, MPX-32 suspends for one second between tries.
- continues to suspend the task and retry until either the resource is free, or the specified number of retries is reached.

  Use the SYSGEN DPTRY directive to specify the number of times MPX-32 tries to allocate a user level resource.

  Setting DPTRY to 1 causes MPX-32 to issue an immediate denial when the resource has the multiprocessor lock set. Setting DPTRY to 0, or failing to specify a value for DPTRY causes MPX-32 to repeat the try/suspend mechanism until the resource is allocated.

  MPX-32 retries indefinitely critical file system structures (root directory, SMAP, DMAP directories) until the resource is allocated.

When access to a multiprocessor resource is denied due to assignment access mode or usage incompatibilities with another task, the requesting task is enqueued or suspended as appropriate (provided that DPTRY has indicated that it will wait for the resouce to become available). If the incompatibility is due to a task in the same CPU, the requesting task is enqueued. If the incompatibility is due to a task in the other CPU, the requesting task is suspended. If suspended, the try/suspend cycle as for multiprocessor locks is performed.

The following conditions can determine if a task can be queued rather than suspended for a multiprocessor resource:

1. The resource is exclusively locked, and the lock owner is in the same system environment as the requesting task.
2. Incompatible access modes are encountered on an implicitly shared resource in which the writer is known to be in the same system environment as the requesting task.
3. A synchronous resource lock cannot be obtained because the lock is owned by another task in the same system environment.

4.13.3 Mounting Multiprocessor Volumes

Mounting a multiprocessor volume is signified by the presence of SYSID in the mount request. The format for the mount request of a multiprocessor volume is:

MOUNT *volname* **ON** *devmnc* **SYS**ID=[**MP***n* | **DP***x*]

n is 0 through F

x is 0 or 1

Multiprocessor volumes can be mounted as public or nonpublic. However, the system volume cannot be mounted as a multiprocessor volume.

When a multiprocessor volume is mounted, J.MOUNT prompts the operator for permission to perform volume cleanup if the volume descriptor indicates the volume was not previously dismounted. When volume cleanup is performed, no regard is given to access from any other port. All resource descriptors are purged of any software multiprocessor information. Therefore, the operator must ensure the integrity of the mount process from all system environments prior to allowing volume cleanup.

If the operator indicates volume cleanup is not to be performed, and J.MOUNT detects the port associated with the SYSID specification has not been previously dismounted, the following message is displayed on the system console:

```
J.MOUNT - WARNING - VOLUME SHOWS PORT DESIGNATOR MP(DP)n ALREADY ALLOCATED
J.MOUNT - REPLY C TO CONTINUE, A TO ABORT:
```

4.13.4 Multiprocessor Resource Restrictions

Some features of volume management provided by MPX-32 are restricted when applied to multiprocessor resources. This is a result of the additional system overhead associated with the processing of these resources. The following sections describe some of the more significant restrictions and potential conflicts that can apply when resources are shared concurrently from separate system environments.

4.13.4.1 EOF Management

Dynamic end-of-file (EOF) information is not available to tasks sharing a multiprocessor resource from separate system environments. The updated EOF information is not available until the writer closes the resource and the reader reallocates the file. However, tasks sharing multiprocessor resources within the same system environment have access to the full range of EOF management allowed to nonmultiprocessor resources. In this context, writer means any task accessing an implicity or explicitly shared resource in write, update or append access mode.

4.13.4.2 EOM Management

Dynamic end-of-medium (EOM) information is not available to tasks sharing a multiprocessor resource from separate system environments. The updated EOM information is not available until the extender has completed the service, and the other task has reallocated the file. Tasks sharing multiprocessor resources within the same system environment have access to the full range of EOM management allowed to nonmultiprocessor resources. In this context, extender means any task accessing an implicitly or explicitly shared resource in update or append access mode, as well as any task requesting a manual extension of an extendible file.

4.13.4.3 Resource Deadlocks

When a task obtains exclusive use of a resource in such a way that it will not (or cannot) release it, any task waiting to gain access to that resource is indefinitely postponed. This situation results whenever a system failure occurs while the multiprocessor RD lock has been set for a task accessing a multiprocessor resource from that system. This usually is the situation when a task (attempting to gain access to a multiprocessor resource in the operational system) appears to be cycling between a suspended and ready-to-run state for an extended period of time.

Under these circumstances, the multiprocessor lock remains in effect until the volume is remounted. This multiprocessor lock can be removed by using the OPCOM UNLOCK directive from the operational system. Once J.UNLOCK has completed, the volume can be remounted from the failed system, but volume cleanup must be inhibited.

4.13.4.4 Reserve/Release Multiported Disk Services (M.RESP/M.RELP)

The multiprocessor features of MPX-32 do not use the reserve and release multiported disk services. The M.RESP service causes the drive to be exclusively reserved to the system environment from which the request was issued. The disk remains reserved until explicitly released by the M.RELP service. If a volume is mounted on the multiported disk drive at the time of a reserve request, it is inaccessible from the other system environment until explicitly released.

MPX-32 performs an implicit reserve of the multiported disk that remains effective for the duration of the IOCL used to set or release the software lock in the appropriate resource descriptor.

The M.RESP and M.RELP services should not be used with the multiprocessor features of MPX-32, or the result can be unpredictable system behavior.

4.13.5 Optimum Use of Multiprocessor Resources

As a result of the restrictions imposed on multiprocessor resources, the following steps can be taken on resources shared by tasks in separate system environments:

1. Do not rely on dynamic EOF management. When creating multiprocessor files, specify EOF management is not in effect. For example:

   ```
   CREATE F filename SIZE=nn EOFM=N
   ```

   This sets EOF to the size of the file (EOM). In this way, concurrent access by multiple tasks does not result in EOF detection until the physical EOF is reached.

2. Have the sharing tasks assign the file for explicit shared use. For example:

   ```
   (Task 1) ASSIGN LFC TO filename ACCESS=(R)   SHARED=Y
                          and
   (Task 2) ASSIGN LFC TO filename ACCESS=(W)   SHARED=Y
   ```

   This allows concurrent access to the file by the reader and writer. Synchronization can then be performed by the record structure in the file (which results in less I/O overhead) or through the synchronous resource lock services provided by MPX-32.

3. To avoid unnecessary I/O overhead associated with multiprocessor resources, do not direct the creation of temporary or swap files on a multiprocessor volume unless absolutely necessary. If the current working volume is a multiprocessor volume, then additional I/O overhead is associated with processing SLO, SBO, and SGO files on the volume.

4. Whenever a system failure occurs in one system, activate J.UNLOCK from the running system. After J.UNLOCK completes, the failed system can be rebooted. With volume cleanup inhibited, remount the volume from the port where the system failure occurred.

5 Resource Assignment/Allocation and I/O

5.1 Introduction

This chapter is an overview of the user interfaces provided for task resource assignment/allocation and the subsequent I/O services available. It assumes the reader is familiar with the terms and concepts presented in Chapter 4.

The MPX-32 Resource Management Module (H.REMM) performs all operations necessary to obtain the physical resources required to execute a task. The MPX-32 I/O control system (IOCS) receives and processes device-independent I/O requests from both user tasks and MPX-32.

The following sections describe the MPX-32 I/O concepts and conventions of:

- logical, wait, no-wait, and device-dependent I/O
- error processing and status posting
- interfaces among IOCS, standard device handlers, and MPX-32
- special system file handling
- file control block (FCB) and type control parameter block (TCPB) setup
- I/O services available to MPX-32 users

5.2 MPX-32 Logical I/O (Device-Independent)

MPX-32 provides versatile logical device-independent I/O capabilities. The user can code references to logical files and request an MPX-32 IOCS to perform I/O.

Several important advantages are gained by performing logical file I/O:

- the user need not be aware of specific device handling requirements
- unprivileged tasks can perform I/O (the I/O instructions are part of the privileged instruction set)
- tasks that perform logical I/O are easier to debug and modify

To provide MPX-32 with sufficient information to create the necessary linkages between the user's logical files and the actual peripheral devices or disk files, the user must:

- identify logical files with logical file codes
- describe logical file attributes with FCBs
- associate logical files with their target physical devices or disk files with logical file code assignments

5.2.1 Logical File Codes

Logical file codes (LFCs) are user defined 1- to 3-character ASCII codes that identify logical files within tasks.

Logical file codes are configured into corresponding FCBs. Refer to the Setting Up File Control Blocks for Device Independent I/O section in this chapter.

5.2.2 File Control Blocks

An FCB must be set up by the user to describe each logical file within a task, and to describe certain attributes of each logical I/O operation.

Information collected by IOCS following each I/O operation is made available to the user by the corresponding FCB. Space in the FCB is reserved for use by IOCS.

The Setting Up File Control Blocks for Device Independent I/O section in this chapter describes the FCB format.

5.2.2.1 Logical I/O Initiation

To initiate a logical I/O operation, users must code into their task a call to one of the data transfer or device access services, accompanied by the address of a corresponding FCB.

5.2.3 Assignment vs. Allocation

The attachment of a task to a resource progresses through two phases: assignment and allocation. The current phase of a particular resource attachment depends on the amount of information supplied by the requestor up to that time.

Assignment is the process of associating an LFC with a system resource. This action informs the system of a task's intention to use a resource, but does not describe the usage (exclusive use, explicit shared, implicit shared) or the intended access mode (read, write, modify, update or append). Hence, the resource is still susceptible to allocation by other tasks, and no guarantee is made that the assigning task can obtain the resource in any specific usage or access mode.

Allocation is the process of securing a resource for a specific usage and access mode for the requesting task. At this point, the task has defined all of its intentions and can perform logical I/O operations on the resource for the usage and access requested.

When an LFC is assigned to a system resource, the task can indicate the mode in which it intends to use the resource (exclusive or explicit shared). If this is done, the assignment becomes an allocation because these usage modes imply that the task is allowed any access mode authorized to it by the resource creator. The task is guaranteed access to the resource when logical I/O is initiated. If a usage mode is not indicated at LFC assignment, implicit shared use of the resource is assumed by default.

When an LFC is attached to a resource with implicit shared use, the resource is not allocated until a specific access mode is indicated. If this occurs at LFC assignment, the assignment becomes an allocation or is deferred until the resource is opened. In the latter case, the resource is not allocated until it is opened, and there is no guarantee the specific access mode is obtained because other tasks may have allocated the resource for implicit shared use in an incompatible access mode.

Refer to Table 5-1, Assign/Open Resource Allocation Matrix.

Table 5-1
Assign/Open Resource Allocation Matrix

Usage specified at Assign	Usage specified at Open	Point when access is specified	Is resource allocated at Assign?	Allocation action at Open
Exclusive	Exclusive	N/A	Yes	None
	Explicit Shared	N/A	Yes	Reallocate and Dequeue
	Implicit Shared	N/A	Yes	None
Explicit Shared	Exclusive	N/A	Yes	Deallocate and Allocate*
	Explicit Shared	N/A	Yes	None
	Implicit Shared	N/A	Yes	None
Implicit Shared	Exclusive	Assign	Yes	Deallocate and Allocate*
	Exclusive	Open	No	Allocate*
	Exclusive	Assign	Yes	Deallocate and Allocate*
	Exclusive	Open	No	Allocate*
	Implicit Shared	Assign	Yes	None
	Implicit Shared	Open	No	Allocate*

* No guarantee that a specific access mode (read, write, modify, update, append) is available at open.

5.2.4 Logical File Code Assignment

Before executing a logical I/O request, the task must associate the appropriate LFC with the target peripheral device or disk file. This is accomplished by LFC assignment.

At this time, the requestor can specify one or more access modes that apply to this assignment. This set of access modes defines the set of allowable I/O operations that can be performed on the resource for the duration of this assignment (read, write, modify, update, and append). The access modes specified at assignment must not allow more access than that allowed to this user by the resource creator or the assignment will be denied. If no access modes are specified, the default access modes in the resource descriptor for this user class are allowed.

A specific resource usage (exclusive or explicit shared) may also be declared at LFC assignment. If not supplied, the resource is assumed to be assigned for implicit shared use. In this case, the resource is not allocated to the requesting task at assignment, unless only one access mode is allowed.

LFCs can be assigned to specific peripheral devices or files when a task is cataloged (static assignment) or during task execution by the M.ASSN service (dynamic assignment).

For tasks that run under TSM control (interactive or batch), static assignments can also be made by the user at run time. For such assignments, if the LFC matches one assigned at catalog time, it replaces the cataloged assignment. If the file code assigned at run time does not match any cataloged assignment, it is added to the cataloged assignments.

Dynamic assignments cannot override cataloged or run-time assignments, and any attempt to do so is treated as an error. To accomplish dynamic override, the user task must first deallocate (deassign) the static assignment by the M.DASN service.

The maximum number of assignments is 245. There is additional space reserved for assignments needed by the operating system.

5.2.4.1 Making Assignments via Resource Requirement Summary (RRS)

The resource requirement summary (RRS) is a structure that defines the assignment requirements of a resource to the Resource Management Module (H.REMM). It is supplied by the Cataloger, LINKX32, or TSM for static assignment of resources to a task or as an argument for the dynamic assignment of a particular resource.

There are distinct types of RRS entries recognized by REMM corresponding to the resource modes and allocation mechanisms available. RRS entries are variable length structures with the first four words generally common for all entries, and the remaining number of words dependent on the RRS type. RRS entries always begin on a doubleword boundary. They must contain an even number of words for static assignments made by parameter task activation or load module activation where pathnames are applied. RRS entries are presented to H.REMM in the following formats.

Unless specified, the first four words of an RRS entry contain the following:

Word 0	Byte 0	cleared
	Bytes 1,2,3	contain a 1- to 3-character, left-justified, blank-filled LFC
Word 1	Byte 0	specifies the RRS type with the following value significance:

Value	Description
1	assign by pathname (RR.PATH)
2	assign to temporary file (RR.TEMP)
3	assign to device (RR.DEVC)
4	assign to LFC (RR.LFC2)
5	assign by segment definition (RR.SPACE)
6	assign by resource ID (RR.RID)
7-8	reserved for future use
9	mount by device mnemonic (RR.MTDEV)
10	assign to ANSI labeled tape (RR.ANS)
11	assign to shadow memory (RR.SHRQ)
12-255	reserved

	Byte 1	specifies the size of this RRS entry in words
	Bytes 2 and 3	vary depending on the RRS type

Restriction: A value must be specified in the RRS type field. There are no defaults applied to this portion of the RRS.

Word 2 Access specification field specifies the access restrictions to be applied to the allocation of this resource. The bit interpretations are as follows:

Bit	Meaning if Set
0	allow read access (RR.READ)
1	allow write access (RR.WRITE)
2	allow modify access (RR.MODFY)
3	allow update access (RR.UPDAT)
4	allow append access (RR.APPND)
5-15	reserved
16	explicit shared use requested (RR.SHAR)
17	exclusive use requested (RR.EXCL)
18	assign as volume mount device (RR.MNT)

Restrictions:

1. The bit pattern specified in bits 0 to 4 must not allow more access than specified in the resource descriptor for this user.
2. Only one of bits 16 to 17 can be set to indicate the intended usage mode. Successful allocation of a resource for exclusive use implies the setting of an exclusive resource lock on that resource.

Defaults: If the access specification field is zero, the default access contained in the resource descriptor for this user is used, and the resource is allocated for implicit shared use.

Word 3 Options specification field specifies the allocation options that are in effect for this assignment. The bit interpretations are as follows:

Bit	Meaning if Set
0	treat as SYC file (TSM/JOB only) (RR.SYC)
1	treat as SGO file (TSM/JOB only) (RR.SGO)
2	treat as SLO file (RR.SLO)
3	treat as SBO file (RR.SBO)
4	explicit blocked I/O (RR.BLK)
5	explicit unblocked I/O (RR.UNBLK)
6	inhibit mount message (RR.NOMSG)
7	reserved for system use
8	automatic open requested (RR.OPEN)
9	user buffer address to be supplied at open (RR.BUFF)
10-12	reserved for system use
13	mount as a public volume (RR.PUBLC)
14	by H.VOMM for special case handling of VOMM assignments (RR.VOMM)
15	spool file when deallocated (RR.SEP)
16	mount as ANSI tape (RR.ANSI)
17-31	reserved

Word 4-*n* RRS type dependent.

Type 1 (Assign by pathname)

Syntax

$ASSIGN *lfc* **TO** *pathname*

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	Plength (RR.PLEN). See Note 1.	Reserved. See Note 2.
2	Access (RR.ACCS)			
3	Options (RR.OPTS)			
4-*n*	Pathname (variable length) (RR.NAME1). See Note 3.			

Notes:

1. RR.PLEN contains character count of pathname/pathname block
2. Byte 3 is zero. This field is used by MPX-32 for big blocking buffers.
3. RR.NAME1 is the resource pathname or pathname block (PNB). Refer to this chapter's H.VOMM Conventions section for format.

Type 2 (Assign to temporary file)

Syntax

$ASSIGN *lfc* **TO TEM**P [=(*volname*)]

Temporary files created in this manner cannot be made permanent unless they are created on a volume which has a valid directory established for this user.

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	Initial file size (RR.PLEN). See Note 1.	
2	Access (RR.ACCS)			
3	Options (RR.OPTS)			
4-7	Volume name (RR.NAME1). See Note 2.			

Notes:

1. RR.PLEN contains the initial file size in logical blocks. If a size is not supplied, the system default is used. Refer to this chapter's Temporary File Assignments section.
2. RR.NAME is a 1- to 16-character left-justified, blank-filled volume name. This element of the RRS entry is optional. If supplied, the temporary file will be created on the specified volume. Otherwise the file will be created on the task's current working volume or any available public volume.

Type 3 (Assign to device)

Syntax

$ASSIGN *lfc* **TO DEV**ICE=*devmnc*

	0 1	7 8	15 16 17	23 24	31
Word 0	Zero	Logical file code (RR.LFC)			
1	Type (RR.TYPE)	Size (RR.SIZE)	Density (RR.DENS). See Note 1.	Zero	
2	Access (RR.ACCS)				
3	Options (RR.OPTS)				
4	Device type (RR.DT3). See Note 2.	Volume number (RR.VLNUM). See Note 3.		Channel number (RR.CHN3). See Note 4.	Subchannel number (RR.SCHN3). See Note 5.
5	Unformatted ID (RR.UNFID). See Note 6.				

Notes:

1. RR.DENS contains an optional density specification for XIO high speed tape units. When specified, this field has the following bit significance:

Bit	Meaning if Set
0	indicates 800 bpi Nonreturn to Zero Inverted (NRZI)
1	indicates 1600 bpi Phase Encoded (PE)
6	indicates 6250 bpi Group Coded Recording (GCR)

If this field is zero, 6250 bpi is set by default.

2. RR.DT3 contains the device type code associated with this device. See Table 5-2. The device type code is the only required portion of this word. If a channel or subchannel is supplied, bits 0 and 16 are set respectively to indicate the presence of these portions of the device address. The Get Device Type Code (M.DEVID) service can be used to obtain the correct value for byte 0 from the appropriate device mnemonic.
3. RR.VLUM contains the volume number for multivolume media.
4. RR.CHN3 contains the logical channel number associated with this device.
5. RR.SCHN3 contains the logical subchannel to be applied with the logical channel number.
6. RR.UNFID is a 1- to 4-character identifier to be associated with an unformatted medium (magnetic tape, disk, or floppy disk). If nonzero, this name appears on the mount/dismount messages; otherwise, SCRA is used as the default identifier.

Type 4 (Assign to LFC)

Syntax

$ASSIGN *lfc* **TO LFC=***lfc*

An LFC to LFC assignment inherits all the access and assignment restrictions specified by the original LFC assignment. The allowable access modes and blocking status cannot be changed by assigning a second LFC to a resource.

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	Zero	
2	Zero	Logical file code (RR.SFC). See Note 1.		
3	Options (RR.OPTS). See Note 2.			

Notes:

1. RR.SFC contains a 1- to 3- character, left-justified, blank-filled LFC. This LFC must have been previously assigned to a resource.
2. RR.OPTS is the options specification field. Automatic OPEN may be selected by specifying the option in this field.

Type 5 (Assign by segment definition)

Dynamic allocation only

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	UDT index (RR.UDTI)	Zero
2	Access (RR.ACCS)			
3	Options (RR.OPTS)			
4	Starting block number (RR.STBLK). See Note 1.			
5	Number of blocks (RR.NBLKS). See Note 2.			

Notes:

1. RR.STBLK is the starting block address of segment definition.
2. RR.NBLKS is the number of contiguous blocks in definition. This type of RRS is only valid for a contiguous volume resource. Extendible files must be allocated by other means.

Restriction: Allocation by segment definition is a privileged assignment mode. An error condition is generated if this form of allocation is attempted by an unprivileged task.

Type 6 (Assign by resource ID)

Syntax

$ASSIGN *lfc* **TO RID=(***resid***)**

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	Zero. See Note 1.	Reserved
2	Access (RR.ACCS)			
3	Options (RR.OPTS)			
4-7	Volume name (RR.NAME1). See Note 2.			
8	Binary creation date (RR.DATE). See Note 3.			
9	Binary creation time (RR.TIME). See Note 4.			
10	Resource descriptor block address (RR.DOFF). See Note 5.			
11	Reserved		Resource type (RR.RTYPE). See Note 6.	

Notes:

1. Word 1, byte 2 is zero. This field is used by MPX-32 for big blocking buffers.
2. RR.NAME1 is a 1- to 16-character, left-justified, blank-filled volume name.
3. RR.DATE is the binary creation date.
4. RR.TIME is the binary creation time.
5. RR.DOFF is the block address of resource descriptor.
6. RR.RTYPE is the resource type value (right-justified).

Type 7 Reserved for Future Use

Type 8 Reserved for Future Use

Type 9 (Mount by device mnemonic)

Syntax

MOUNT *volname* **ON** *devmnc*

If the public volume option is selected, the volume is mounted for public use. Refer to this manual's Chapter 6 for a description of the dynamic assignment/allocation services M.ASSN and M.DASN.

	0 1 7	8 15	16 17 23	24 31
Word 0	Zero	System ID (RR.SYSID). See Note 1.		
1	Type (RR.TYPE)	Size (RR.SIZE)	Zero	
2	Access (RR.ACCS)			
3	Options (RR.OPTS)			
4-7	Volume name (RR.NAME1). See Note 2.			
* 8	Device type (RR.DT9)	Reserved	Channel number (RR.CHN9)	Subchannel number (RR.SCHN9)
9	Zero. See Note 3.			

Notes:

1. RR.SYSID is a 3-character SYSID for dual/multi port volumes.
2. RR.NAME1 is a 1- to 16-character, left-justified, blank-filled volume name.
3. Word 9 is zero (reserved for system use).

* Word 8 is the device specification word. Format is the same as for type 3, word 4. This element describes the device where the volume is to be mounted. If a complete address is specified, an attempt is made to mount the volume on that device only. Otherwise, the volume is mounted on any device that matches the portion of the specification word supplied in the RRS.

Type 10 (Assign to ANSI tape)

Syntax

$ASSIGN *lfc* **TO @ANSITAPE(***lvid***)** *filename*

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE)	Size (RR.SIZE)	Format (RR.FORM). See Note 1.	Protect (RR.PROT)
2	Access (RR.ACCS). See Note 2.			
3	Options (RR.OPTS)			
4	Record length (RR.RECL)		Block size (RR.BSIZE)	
5	Generation number (RR.GENN)			
6	Generation version number (RR.GENV)			
7	Absolute termination date (RR.EXPIA). See Note 3.			
8	Relative termination date (RR.EXPIR). See Note 3.		Logical volume identifier (RR.LVID)	
9	RR.LVID (cont.)			
10-13	17-character file identifier (RR.AFID)			
14	RR.AFID (cont.)	Reserved		
15	Reserved			

Notes:

1. RR.FORM values are as follows:

Value	Meaning
Numeric 0	use default format
ASCII F	fixed length record format
ASCII D	variable length record format
ASCII S	spanned record format

2. RR.ACCS accesses a resource in the read, write, update, or append mode. The access modes are mutually exclusive.
3. For the absolute termination date, word 7 is one blank followed by a 2-digit number that specifies the year and the first digit of the 3-digit day. Word 8, bytes 0 and 1 contain the remaining two digits of the day.

   For the relative termination date, word 7 is zeroed; word 8, bytes 0 and 1 contain the binary relative termination date.

Type 11 (Assign to shadow memory)

	0 7	8 15	16 23	24 31
Word 0	Zero			
1	Type (RR.TYPE)	Size (RR.SIZE)	Shadow flags (RR.SHAD). See Note 1.	
2	Start address (RR.SADD). See Note 2.			
3	End address (RR.EADD). See Note 3.			

Notes:

1. RR.SHAD contains the shadow flags that qualify the start and end addresses, or specify what portions of the task are to be shadowed. The bit interpretations are as follows:

Bit	Meaning if Set
0-7	reserved
8	shadow the task (RR.SHTSK)
9	shadow the TSA (RR.SHTSA)
10	shadow the stack (RR.SHST)
11	shadow memory is required (RR.SHRQ)
12	shadow the entire task (RR.SHALL)
13	absolute address (RR.ABS)
14	relative to the code section origin (RR.CREL)
15	relative to the data section origin (RR.DREL)

2. RR.SADD is the starting logical byte address of the memory space to be put in shadow memory. The address is considered relative to the start of the task (T.BIAS), unless qualified with shadow flags in word 1.
3. RR.EADD is the ending logical byte address of the memory space to be put in shadow memory. The address is considered to be relative to the start of the task (T.BIAS), unless qualified with shadow flags in word 1.

The start and end addresses are specified as a byte address in the TSM SHADOW directive and the RRS. The range of address space specified is inclusive. Internal to MPX-32, these addresses are map block bounded. Due to the map block granularity of 2KW, more address space than requested can be shadowed, but never less.

5.2.4.2 Temporary File Assignments

Temporary files are created explicitly by the Create Temporary File (M.TEMP) service and assigned by the resource identifier (RID) obtained when the file was created.

Temporary files are implicitly created as the result of an LFC assignment, such as

```
ASSIGN OUT TO TEMP=(volume)
```

When a temporary file is created in this manner, the resource is given the following characteristics:

- All access modes are allowed.
- The file is shareable.
- The file is automatically and manually extendible.
- The initial file size is 16 blocks, unless a size is specified in the RRS.
- File extensions are made in a minimum of 32 block increments.

If blocked or unblocked is not specified as an assignment option to a temporary file, all I/O is blocked by default.

5.2.5 Opening a Resource for Logical I/O

Before any service requested by the user to initiate an I/O operation is completed, the user's logical file must be opened. This operation establishes the access mode for subsequent I/O operations. The specified access mode must have been allowed by the resource assignment or an error occurs.

If an access mode is not specified, the resource is opened for the appropriate default access, unless only a specific access mode was allowed at assignment. In that case, the resource is opened for the allowed access mode. The default access modes are read access for files and update access for peripheral device assignments.

A resource usage mode, exclusively or explicitly shared, is also specified at open. When this is done, the supplied usage overrides that specified at LFC assignment. This results in the attachment to the resource reverting to an assignment, if allocated. If the LFC assignment allocated the resource for a shareable usage, there is no guarantee that the resource is available for reallocation when the usage specification is overridden at open. A previously allocated resource can not be available at open in this case.

The task performs the function directly with the M.OPENR service, automatically during assignment, or automatically by IOCS when an I/O initiation service request is made and the logical file is not open. In the latter case, the default access mode in effect for that assignment applies.

When the logical file is properly opened, the FCB and FAT are linked together to complete the I/O structure for IOCS.

5.3 Resource Conflicts and Error Handling

In the course of processing resource assignments, conditions exist that require additional information from the task requesting the resource. MPX-32 allows the user the option of waiting for resources that are temporarily not available. The requesting task can also dictate the manner in which error or denial status is presented.

The caller notification packet (CNP) is the mechanism used by the Resource Management Module (H.REMM) and the Volume Management Module (H.VOMM) for handling abnormal conditions that result during resource requests. This structure is a standardized parameter optionally supplied to many of the services provided by REMM and VOMM.

The CNP consists of a 5- to 6-word area containing the following information, all or part may be used by the particular service being called:

Word 0 — Wait request time-out value interpreted as follows:

Value	Description
0	place the requesting task in a wait state until the designated service can be completed
>0	return immediately with a denial code if the service cannot be completed
-n	place the requesting task in a wait state until the designated service can be completed or until the expiration of *n* timer units, whichever occurs first

Word 1 — abnormal return address — if an error condition or denial condition occurs, control is transferred back to the task at this address

Word 2 — options field (bytes 0 and 1) — a bit sequence and/or value used to provide additional information that is necessary to fully define the calling sequence for a particular service

status field (bytes 2 and 3) — a right-justified, numeric value identifying the return status for this call

Word 3 — actual file size created

Word 4 — reserved for future use

Word 5 — parameter link — required only for a resource assignment where automatic open has been specified in the option word of the RRS. If specified, this word must contain the address of a valid FCB for this assignment.

5.3.1 Status Posting and Return Conventions

The services that accept a CNP as a calling parameter adhere to a common status posting and return convention. Status is always posted as a numeric value that represents one of three conditions:

- successful — CC1 not set indicates the service completed without any abnormalities.
- error — indicates a condition occurred during execution that precluded any continuation of the service. This condition has a positive numeric value identifying an error condition specific to the service.
- denial — indicates the service was not completed due to resource nonavailability or other system constraints which do not necessarily preclude the continuation of the service at a later time. This condition is also represented by a numeric value specific to the service, but is posted only if the caller has indicated that the task is not to wait for the condition to be alleviated. Denial conditions are returned only by H.REMM.

The following conventions apply to the posting of status and the return sequence applied to service calls that accept a CNP as an input parameter:

- CC1 is set to indicate the posting of a nonzero status value.
- The task is never aborted as the result of posting a status value.
- The location of status and return sequence for denials and errors is dependent on the presence or absence of a CNP:

  If CNP is supplied:
  - Status is posted in the status field of the CNP.
  - For all error conditions, the return is made by the abnormal return address if supplied. Otherwise, a normal return occurs.
  - For all denial conditions, the task is enqueued if requested; otherwise, the return sequence is the same as an error condition.

  If CNP is not supplied:
  - Status is posted in R7.
  - For all error conditions, a normal return occurs.
  - For all denial conditions, the task is enqueued until the service can be completed.

The following is a summary of the status codes returned by H.REMM for resource allocation:

Value	Description
0	successful completion
1	unable to locate resource (invalid pathname or memory partition definition)
2	specified access mode not allowed
3	FPT/FAT space not available
4	blocking buffer space not available
5	shared memory table (SMT) entry not found
6	volume assignment table (VAT) space not available
7	static assignment to dynamic common
8	unrecoverable I/O error to volume
9	invalid usage specification
10	dynamic partition definition exceeds memory limitations
11	invalid resource requirement summary (RRS) entry
12	LFC logically equated to an unassigned LFC
13	assigned device not in system
14	resource already allocated by requesting task
15	SGO or SYC assignment by real-time task
16	common memory conflicts with task's address space
17	duplicate LFC assignment attempted
18	invalid device specification
19	invalid resource ID (RID)
20	specified volume not mounted
21	J.MOUNT run request failed
22	resource is marked for deletion
23	assigned device is marked off-line
24	segment definition allocation by unprivileged task
25	random access not allowed for this access mode
26	user attempting to open SYC file in a write mode
27	resource already opened by this task in a different access mode
28	invalid access specification at open
29	specified LFC is not assigned to a resource for this task
30	invalid allocation index
31	close request issued for an unopened resource
32	attempt to release an exclusive resource lock that was not owned by this task, or a synchronous lock that was not set
33	attempt to release an exclusive resource lock on a resource that has been allocated for exclusive use
34	attempt to mount a public volume without the system administrator attribute
35	attempt to exclude memory partition that is not mapped into requesting task's address space

Value	Description
36	physical memory already allocated
37	invalid J.MOUNT request
38	time out occurred while waiting for resource to become available
39	unable to perform write back
40	invalid load module
41	invalid physical address specified
42	user requested abort of mount process
43	user requested hold on mount process
44	writeback requested and shared image has no writeback section
45	loading error during inclusion of read-only section of shared image
46	unable to obtain resource descriptor lock (multiport only)
47	loading error during inclusion of read/write section of shared image
48	incompatible load address for shared image
49	excessive multicopied shared images with no read only section

Status codes in the range 50 to 63 represent denial conditions that result in suspension of the task if a CNP is not supplied or enqueue is indicated by the time-out value in the CNP:

Value	Description
50	resource is locked by another task
51	shareable resource is allocated by another task in an incompatible access mode
52	volume space is not available
53	assigned device is not available
54	unable to allocate resource for specified usage
55	allocated resource table (ART) space is not available
56	task requires shadow memory and none is configured
57	volume is not available for mount with requested usage
58	shared memory table (SMT) space is not available
59	mounted volume table (MVT) space is not available
60	resource descriptor space definition conflict
61	unable to locate or retrieve resource descriptor
62-63	reserved
64	task's DSECT space requirements overlap the task's task service area (TSA) space requirements
65	task's DSECT space requirements overlap the task's CSECT space requirements or, if no CSECT, load module is too large to fit in user's address space
66	software checksum. Error may be fixed by recataloging.
67	excessive memory request
68	excessive volume space requested
69	invalid user name specified

Value	Description
70	invalid privileged activation
71	reserved
72	unable to resume SYSINIT on tape activation
73	file overlap. Check system console.
74	loading error
75	invalid work volume/directory
76	user attempted deallocation of TSA
77	a task destroyed the allocation linkages in it's dynamic expansion space
78	unable to load debugger with task
79	invalid caller notification packet (CNP) address
80	shared image version level is not compatible with executable image
81	invalid activation of a base mode task on a system configured for non-base task execution
82	invalid activation of an ADA task on a system configured without ADA support
83	insufficient logical address space to activate task
84	invalid logical position for extended MPX
85	reserved
86	cannot dismount the system volume
87	unable to dismount public volume because compatible mode public volume dismount (CMPMM) option was specified at SYSGEN
88	unable to dismount public volume. SA attribute required
89	public dismount denied due to missing option for public volume in the dismount request
90	reserved
91	unable to mount volume due to pending physical dismount
92-255	reserved

5.4 MPX-32 Volume Resource Access

The MPX-32 Volume Management Module (H.VOMM) maintains a resource descriptor (RD) list on disk for each mounted volume in the system. A descriptor exists for every currently active volume resource (permanent files, temporary files, directories and memory partitions). Each resource descriptor contains all the access, accounting, and space definition information pertaining to the associated resource.

When a volume resource is assigned for I/O, H.REMM reads the associated resource descriptor. The usage and access specifications supplied at assignment are verified against those allowed by the resource descriptor for protection and resource integrity. If a discrepancy exists, the task is aborted (static assignment) or the appropriate error code is returned (dynamic assignment). In either case, the assignment is denied.

The description of a valid volume resource is placed in the file assignment table (FAT). IOCS uses the FAT entry to map the appropriate disk and file. Access to disk files is sequential, starting with the first relative block in the file, unless the user has indicated random access in the FCB. However, random access is not allowed in the write and append access modes.

Permanent files are accessed by pathname or resource identifier (an 8-word unique identifier obtained when the resource was created). Temporary files are accessed by resource identifier only.

5.4.1 Volume Resource Space Management

The MPX-32 Volume Management Module (H.VOMM) provides space management for all currently mounted volumes in the system. Disk space is allocated to files from the available space not dedicated to volume management, an area whose size is determined when the volume is formatted. Volume space is allocated from this area of the disk in units, each consisting of an integral number of 192 word blocks. Refer to this chapter's Allocation Units section. MPX-32 prevents any task from performing asynchronous aborts and deletes while a service that modifies the allocation of disk space is acting on that task. Any such request is deferred until the service completes.

5.4.2 Temporary vs. Permanent Files

Temporary files have all the access and protection attributes associated with permanent files. However, temporary files are automatically deallocated, and their volume space is returned to the pool of available space when a task exits or aborts.

Temporary files created by the Create Temporary File (M.TEMP) service can be shared by tasks that received the associated resource identifier (RID) from the creating task. In this case, the volume space allocated to the temporary file is not deallocated until the last task assigned to it deallocates the file or exits.

Temporary files are made permanent if they are created on a specific volume in a valid directory for the user making the request.

Permanent files and memory partitions remain defined on the volume until they are explicitly deleted by a user who has access to them.

5.4.3 System Directory

The system directory is a special directory that resides on the system volume. It is identified within the pathname structure by the keyword SYSTEM, a name reserved for MPX-32.

All executable images or load modules with the system administrator attribute must reside in the system directory.

Any other files created on a volume can reside in the system directory if desired. However, no special significance is given to these files.

5.5 MPX-32 Device Access

When a device is assigned for I/O, H.REMM verifies that the device is available, allocates blocking buffers required for blocked I/O to disk, magnetic tape, or floppy disk, and identifies the device for IOCS in the FAT. If a device is not available (not included in the SYSGEN configuration of a system, off-line, etc.), H.REMM aborts the task (static assignment) or returns an error code (dynamic assignment).

Throughout the reference manual, the generic descriptor *devmnc* indicates that a device can be specified.

MPX-32 specifies device addresses using a combination of three levels of identification: device type, device channel/controller address, and device address/subaddress. The combinations identifying device addresses include the following:

- To allocate the first available device (of the type requested) specify the generic device type mnemonic only.
- To allocate the first available device (of the type requested) on a specific channel or controller specify the generic device type mnemonic and the channel/controller address.
- To allocate a particular device specify the device type mnemonic, channel/controller, and address/subaddress of the device.

5.5.1 Magnetic Tape

A multivolume magnetic tape is a set of one or more physical reels of magnetic tape (255 maximum) processed as a continuous reel. Multivolume magnetic tape processing is supported under MPX-32 by the Multivolume Magnetic Tape Management Module (H.MVMT).

The multivolume magnetic tape can be used to transfer data to any MPX-32 Revision 2 or 3 system (refer to Table 5-2). It must be used when data to be transferred cannot fit on a single tape.

Table 5-2
Multivolume Magnetic Tape Data Transfers Between Different Operating Systems

MPX-32 Source System Revision*	MPX-32 Destination System Revision**	Result
Pre-3.3	Pre-3.3	Possible data loss at end of tape
Pre-3.3	3.3 or later	H.MVMT treats magnetic tape as pre-3.3, possible data loss at end of tape
3.3 or later	Pre-3.3	The pre-3.3 H.MVMT treats the magnetic tape as pre-3.3, possible data loss at end of tape.
3.3 or later	3.3 or later	H.MVMT detects the new header and expects a trailer.

* The source system generates the multivolume magnetic tape.

** The destination system reads the multivolume magnetic tape.

Multivolume tape reels have the following format:

192 Word header	Data	End of tape marker	Remaining data	End of file marker	192-word trailer

The 192-word header is a tape label produced as the first record for each volume of a multivolume file. The volume number and the reel identifier portions of this record are verified during subsequent read operations. The header has the following format:

Words	Contents
0	reel ID (4-character ASCII entry)
1	volume number 1 to 255 in binary format
2-3	date (ASCII) in Gregorian format (*mm/dd/yy*)
4	time (byte binary) as follows:

Byte	Meaning
0	hour
1	minutes
2	seconds
3	intervals (1/60th of a second)

5	multivolume revision number (ASCII) (0001)
6-7	'HEADER␀␀'
8-9	'FOR␀MULT'
10-11	'IVOLUME␀'
12-13	'MAGNETIC'
14-15	'␀TAPES␀F'
16-17	'OR␀MPX-3'
18-19	'2␀␀␀␀␀␀␀'
20-191	reserved (zero)

The format for the trailer is as follows:

Words	Contents
0	reel ID (4-character ASCII entry)
1	volume number 1 to 255 in binary format
2-3	date (ASCII) in Gregorian format (*mm/dd/yy*)
4*	time (byte binary) as follows:

Byte	Meaning
0	hour
1	minutes
2	seconds
3	intervals (1/60th of a second)

5	multivolume revision number (ASCII) (0001)
6-7	'TRAILER␣'
8-9	'FOR␣MULT'
10-11	'IVOLUME␣'
12-13	'MAGNETIC'
14-15	'␣TAPES␣F'
16-17	'OR␣MPX-3'
18-19	'2␣␣␣␣␣␣␣'
20-191	reserved (zero)

* The date and time stored in the trailer is when the trailer was built; it indicates the date and time stored in the header and should be identical to the date and time of the header.

Multivolume magnetic tape processing mode is invoked whenever an assignment to a magnetic tape unit is encountered where multivolume is indicated by a volume number.

Multivolume processing is automatic for magnetic tape operations in the forward direction, such as read, write, advance, and erase. For reverse direction operations, such as rewind or backspace, the user is required to provide processing logic for proper reel manipulation and positioning requests.

Volume numbers in the range of 1 to 255 are provided as an operations aid in mounting and dismounting physical reels of magnetic tape. MPX-32 treats the volume numbers in circular fashion (for example, volume 1 follows volume 255 in the numbering scheme).

A scratch tape (SCRA) is not assigned if the multivolume mode is specified. Also, a unit applicable for multivolume magnetic tape operations cannot be designated a shared (SHR) device.

For multivolume magnetic tape processing, MPX-32 cannot pass end-of-medium (EOM) indicators to the user by the FCB. If a read, write, or advance operation is requested when tape is at end-of-medium, the system issues a dismount message to the console teletypewriter for the current volume (reel), followed by a mount request for the next sequential volume number before performing the requested operation.

Depending on the MPX-32 revision of the source system, read and write requests complete the following for a multivolume magnetic tape:

Source system*	Request	Result
Pre-3.3	Read	No check for expanded header
	Write	No expanded header or trailer exists
3.3 or later	Read at BOT	Checks for expanded header. If expanded header is detected, bit 2 of DFT.FLGS is set.
	Write	Writes expanded header and trailer
* The source system generates the multivolume magnetic tape.		

The system does not recognize multivolume mode specification when a magnetic tape is positioned at load point and a rewind or backspace operation is requested.

For Volume Manager restore operations, special processing occurs if the file being restored resides on two or more reels of a multivolume magnetic tape.

5.5.2 Unformatted Media

When a task is activated that has a device assignment to magnetic tape, disk, or floppy disk, the resource is treated as an unformatted medium. This implies that a medium must be mounted on the drive prior to the initiation of any I/O operations. The device is treated, in effect, as one continuous file for sequential I/O operations. Table 5-3 lists the physical dimensions of the medium for the various disk drives supported by MPX-32.

If the blocked or unblocked option is not selected at LFC assignment, all I/O to unformatted media is blocked by default.

When the resource is opened for I/O, a mount message is displayed on the operator's console. This message is inhibited, if desired, by the SYSGEN directive or assignment option. A mount message is not issued for assignments to shared devices. The format of the mount message for unformatted media is:

```
MOUNT reel VOL volume ON devmnc
TASK  taskname,taskno REPLY R,H,A OR DEVICE:
jobno
```

reel specifies a 1- to 4-character identifier for the reel. If not specified, the default is SCRA (Scratch).

volume identifies the volume number to mount if multivolume tape

devmnc is the device mnemonic for the tape unit selected in response to the assignment. If a specific channel and subaddress are supplied in the assignment, the specific tape drive is selected and named in the message; otherwise, a unit is selected by the system and its complete address is named in the message.

taskname is the name of the task the unformatted medium is assigned to

taskno is the task number assigned to this task by the system

`R,A,H OR DEVICE`

the device listed in the message can be allocated and the task resumed (`R`), a different device can be selected (`DEVICE`), the task can be aborted (`A`), or the task can be held with the specified device deallocated (`H`). If an R response is given and a high speed XIO tape drive is being used, its density can be changed when the software select feature is enabled on the tape unit front panel. If specified, it overrides any specification made at assignment. Values are:

Value	Description
N or 800	indicates 800 bpi Nonreturn to Zero inverted (NRZI)
P or 1600	indicates 1600 bpi Phase Encoded (PE)
G or 6250	indicates 6250 bpi Group Coded Recording (GCR). Default.

Example usage: `RN`, `R1600`, etc.

Note: Do not insert blanks or commas.

jobno if the task is part of a batch job, identifies the job by job number

Response:

To indicate that the drive specified in the mount message is ready and proceed with the task, mount the appropriate medium on the drive and type `R` (resume), optionally followed by a density specification if the drive is a high speed XIO tape unit. To abort the task, type `A` (abort). To hold the task and deallocate the specified device, type `H` (hold). The task is then resumed by the CONTINUE directive. At this time, a drive is selected by the system and the mount message is redisplayed.

To select a drive other than the drive specified in the message, enter the mnemonic of the drive you want to use. Any of the three levels of device identification can be used. The mount message is reissued. Mount the tape and type `R` (resume) if satisfactory, or, if not satisfactory, type `A` (abort), `H` (hold), or select a device as just described.

Table 5-3
Disk Description Table

SYSGEN Disk Code	FL001	MH080	MH160	MH300	MH340	MH600
Unit Size	Floppy	80MB	160MB	300MB	340MB	600MB
Type	MH	MH	MH	MH	MH	MH
Sector Size	256B*	192W	192W	192W	192W	192W
Sectors/Track	26	20	20	20	20	20
Number of Heads	2	5	10	19	24	40
Maximum Cylinders	77	823	823	823	711	843
Sectors/ Cylinder	52	100	200	380	480	800
Total Sectors	4004	82,300	164,600	312,740	341,280	674,400
Maximum Data Byte Capacity	1.02MB	63.20MB	126.41MB	240.15MB	262.10MB	517.94MB

* Smallest accessible block size=192W

MH=Moving Head
Sectors/Cylinder=(Sectors/Track) x (Number of Heads)
Total Sectors=(Sectors/Cylinder) x (Maximum Cylinders)
Maximum Data Byte Capacity=(Maximum Sectors) x 768 bytes/sector

5.5.3 Examples of Device Identification Levels

Examples of the three types of device specification follow:

Form 1 - Generic Device Class

```
ASSIGN OUT TO DEV=M9 MULTIVOL=1 ID=SRCE
```

In this example, the device assigned to logical file code (LFC) `OUT` is any `9`-track tape unit on any channel. The multivolume reel number is `1`. The reel identifier is `SRCE` and the tape is blocked.

Form 2 - Generic Device Class and Channel/Controller

```
ASSIGN OUT TO DEV=M910 ID=SRCE2 BLOCKED=N
```

In this example, the device assigned to logical file code (LFC) `OUT` is the first available `9`-track tape unit on channel `10`. The specification is invalid if a 9-track tape unit does not exist on the channel. The reel identifier is supplied. The tape is unblocked and is not multivolume.

Form 3 - Specific Device Request

```
ASSIGN OUT TO DEV=M91001
```

In this example, the device assigned to logical file code (LFC) `OUT` is the `9`-track tape unit `01` on channel `10`. The specification is invalid if unit 01 on channel 10 is not a 9-track tape. The tape reel identifier is SCRA. The tape is blocked and is not multivolume.

5.5.4 GPMC Devices

GPMC/GPDC device specifications are similar to the general structure previously described. For instance, the terminal at subaddress `04` on GPMC 01 whose channel address is `20` would be identified as follows:

```
ASSIGN OUT TO DEV=TY2004
```

5.5.5 NULL Device

A special device type, NU, is available for null device specifications. Files accessed using this device type generate an end-of-file (EOF) upon attempt to read and normal completion upon attempt to write.

5.5.6 System Console

LFCs are assigned to the operator console by using the device type CT.

5.5.7 Special File Attributes

There are two special file attributes that can be applied to resources at LFC assignment: print and punch. These attributes are supplied as options on the TSM $ASSIGN statement. For example:

```
$ASSIGN OUT TO OUTPUT PRINT          (permanent file)
$ASSIGN OUT TO TEMP=(volname) PRINT  (temporary file)
```

See the Spooled Output with Print or Punch Attribute section in this chapter for a description of the handling of special file attributes.

5.6 Samples

A description of device selection possibilities are as follows:

Disk

DC	any disk except memory disk
DM	any moving head or memory disk
DM08	any moving head disk on channel 08
DM0801	moving head disk 01 on channel 08
DM0002	memory disk 02 on channel 00
DF	any fixed head disk
DF04	any fixed head disk on channel 04
DF0401	fixed head disk 01 on channel 04
FL70F0	floppy disk 0 on controller F channel 70

Tape

MT	any magnetic tape
M9	any 9-track magnetic tape
M910	any 9-track magnetic tape on channel 10
M91002	9-track magnetic tape 02 on channel 10
M7	any 7-track magnetic tape
M712	any 7-track magnetic tape on channel 12
M71201	7-track magnetic tape 01 on channel 12

Card Equipment

CR	any card reader
CR78	any card reader on channel 78
CR7800	card reader 00 on channel 78

Line Printer

LP	any line printer
LP7A	any line printer on channel 7A
LP7A00	line printer 00 on channel 7A

Table 5-4
MPX-32 Device Type Codes and Mnemonics

Device Type Code	Device Type Mnemonic	Device Description
00	CT	Operator console (not assignable)
01	DC	Any disk unit except memory disk
02	DM	Any moving head or memory disk
03	DF	Any fixed head disk
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 disk
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 disk except memory disk

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

5.7 Device-Independent I/O Processing

A task starts I/O operations (reads, writes, etc.) by issuing service calls to IOCS. IOCS validates the logical address of the task's data buffer (defined in the transfer control word (TCW) of the FCB), and links an I/O request containing the TCW information to a queue for the appropriate device handler. The I/O requests are queued by the software priority (1 to 64) of requesting tasks.

The handler issues appropriate instructions to the device controller (command device, test device, or start I/O).

The controller performs the I/O. For example, it reads a record into the task's data buffer. When the requested I/O is complete, the controller issues a service interrupt (SI).

If the handler is passing the address of a list of directives or data (IOCL) for a controller to operate on, the SI is returned from the controller when all operations specified in the list have been completed.

The processing that occurs when the handler receives the SI interrupt from the controller depends on whether the user has established a wait I/O or no-wait I/O environment by the FCB.

5.7.1 Wait I/O

If wait I/O is indicated in the FCB, the task waits (suspends) until the I/O operation completes. IOCS returns to the task at the point following the I/O service call.

5.7.1.1 Wait I/O Errors

If an error occurs during an I/O operation to tape or disk, the handler automatically retries the operation. If retry operations fail, the handler passes the error to IOCS and IOCS posts status in the FCB (word 3, and optionally words 11 or 12). The task can take action or not as described in the next section. When an error is detected on a card reader, line printer, or other device that does not have automatic retry, and operator intervention is applicable, the handler passes the error to IOCS and IOCS issues a message on the system console indicating the device is not operating:

```
*devmnc INOP: R,A?
```

Sample criteria for the INOP message are: the printer runs out of paper, a card reader or printer malfunctions, etc.

IOCS allows the console operator to correct the condition or abort. If the device is fixed, the operator types `R` (retry). IOCS re-establishes the entry conditions for the handler (passes the TCW with the initial transfer count, etc.) and calls the handler to retry the I/O operation that was in error. The error status is cleared and I/O proceeds normally. If the operation aborts, IOCS starts abort processing.

5.7.1.2 Wait I/O Exit and I/O Abort Processing

If error processing is not applicable (disk or magnetic tape) or if the operator has responded A (abort) to the INOP message, IOCS either aborts the task or transfers control to the task at the address specified in word 6 of the FCB. If the task gains control, it examines the contents of word 3 and optionally words 11 and 12 as applicable, and performs its own exit or abort processing.

If the user has not defined an error return address for wait I/O in the FCB, IOCS aborts the task and displays an abort message on the system console:

I/O ERR DEV: *devmnc* STATUS *statusword* LFC:*lfc mm/dd/yy hh:mm:ss*

IOCS displays status word 3 from the FCB on the system console.

For I/O to XIO devices, a second line is displayed along with the I/O ERR message:

XIO SENSE STATUS = *senseword*

where *senseword* is the sense information returned in FCB.IST1 of an extended FCB.

5.7.1.3 Error Processing and Status Inhibit

The user sets word 2, bit 1 of the FCB to bypass error processing by handlers and IOCS. On error, the status word of the FCB is still set by IOCS (unless bit 3 is set as described below), and IOCS transfers control to the task normally. The task must perform any error processing.

If the user sets word 2, bit 3 of the FCB, he inhibits handlers from checking status in any respect. No error status is returned to IOCS. All I/O appears to complete without error.

5.7.2 No-Wait I/O

If the user has indicated no-wait I/O in the FCB, IOCS returns control to the task immediately after an I/O request is queued. The task continues executing in parallel with its I/O. When the handler fields an SI interrupt for the specified I/O operation, it notifies the MPX-32 executive. The executive links the I/O queue entry to a software interrupt list for the task (a task interrupt). When the task is to gain control, the executive passes control to IOCS. An unprivileged user is limited to five no-wait I/O requests.

If the task issues a second concurrent no-wait I/O request to the same LFC, IOCS immediately places the task in ANYWAIT (prior to starting the I/O) until the first I/O completes end action processing.

5.7.2.1 No-Wait I/O Complete Without Errors

IOCS checks the address specified by the user in the FCB for no-wait I/O end action processing. If I/O completes successfully, it routes control to the address provided by the user for normal end processing (word 13). If the user has not specified this address in the FCB, IOCS returns control to the executive and the task continues executing at the point where the task interrupt occurred.

5.7.2.2 No-Wait I/O Complete with Errors

When IOCS gains control on a task interrupt and an error occurs, IOCS routes control to the task at the user-supplied error return address in word 14 of the FCB. If the user has not supplied this address, control is returned to the executive and then to the task so that it continues execution where the task interrupt occurred.

The FCB (word 3 and optionally words 11 and 12) indicates the cause of an error. The task is responsible for examining the word(s) and for any recovery procedure. IOCS makes no attempt to recover from an error condition through operator intervention when a task uses no-wait I/O.

5.7.2.3 No-Wait End-Action Return to IOCS

A task using no-wait I/O end-action processing should return to IOCS by an SVC 1,X'2C' after normal or error processing is complete. IOCS returns control to the executive, which returns control to the task where the task interrupt occurred. In any end-action processing, R1 points to the FCB address.

5.7.3 Direct I/O

Within a task, the user can temporarily bypass normal IOCS and handler functions by coding a handler and attaching it to a specific channel (service interrupt level). Direct I/O is totally under the user's control and acquires data at rates that prohibit IOCS overhead.

To perform direct I/O, the task passes a TCW directly to a device, completely bypassing the IOCS. The task is responsible for any control structures related to the I/O operation (it does not, for example, have use of a FCB) and it must field interrupts on its own. The user must be familiar with the hardware and its response to software instructions, and must implement all support pertaining to I/O for the device.

Before connecting his own handler, the user issues a reserve channel call to IOCS. IOCS then holds all outstanding or subsequent requests for the specified channel until the task issues a release channel call. When IOCS receives the release request, it resumes normal processing on the channel with the standard handler.

Direct I/O is not the same as developing an I/O handler and linking it into the system at SYSGEN. Direct I/O is a privileged operation.

5.7.4 Blocked I/O

Blocked I/O processing is supported under MPX-32 by the Blocked Data Management Module (H.BKDM).

I/O to disk files and magnetic tapes is blocked or unblocked depending on the device assignment or the setting of word 2, bit 5 of the FCB at open. If the user does not set bit 5, I/O is blocked or unblocked as specified by the device assignment. If bit 5 is set at open, I/O is blocked. The FCB definition overrides any assignment specification of unblocked (specified explicitly at assignment).

For blocked I/O, REMM automatically allocates and uses a 192-word blocking buffer that provides intermediate buffering between a task's data buffer and a device. A block is 192 words long and records that can be packed into the block are a maximum of 254 bytes long. Longer records are truncated. The particular buffer used for blocked I/O can be user-supplied by the FCB at open or allocated from the buffer area of the TSA.

On input, IOCS transfers a block of records from a device into the blocking buffer and moves them into the task's data buffer one logical record at a time. On output, IOCS transfers one logical record at a time from the task's data buffer into the blocking buffer and outputs the accumulated records in 192-word blocks.

Reads and writes to the same blocked file or tape by the same task are not mixed because they interfere with the blocking buffer operations being performed by IOCS. A read attempted while IOCS is writing out a group of records to the blocking buffer (or a write when reading) is not executed and IOCS aborts the task.

Special care must be taken when writing to a blocked file in modify mode. Blocked files are modified on a block-by-block basis rather than record-by-record. MPX-32 clears each block of a blocked file before writing records to the block. The user may rewrite any number of records in the cleared block. However, since MPX-32 does not preread the block before clearing it, the user must preread any records in a block which will be modified and rewrite them, if necessary.

Table 5-5
Assign/Open Block Mode Determination Matrix

Form of Assignment	Open by M.OPENR			Open by M.FILE
	No CNP Options Set	CNP Word 2 Bit 10 Set	CNP Word 2 Bit 11 set	FCB Word 2 Bit 5 set
AS LFC TO *pathname*	Blocked	Unblocked	Blocked	Blocked
AS LFC TO *pathname* BLO=Y	Blocked	Unblocked	Blocked	Blocked
AS LFC TO *pathname* BLO=N	Unblocked	Unblocked	Blocked	Blocked
A1 LFC=*filename*	Blocked	Unblocked	Blocked	Blocked
A1 LFC=*filename*,,U	Unblocked	Unblocked	Blocked	Blocked

Notes:

When bit 10 is set, the resource is opened in unblocked mode, overriding any specification made at assignment.

When bit 5 or bit 11 is set, the resource is opened in blocked mode, overriding any specification made at assignment.

If both bit 10 and bit 11 are set, the resource is opened in blocked mode.

5.7.5 End-of-File and End-of-Medium Processing

Table 5-6 describes the different types of end-of-file (EOF) and end-of-medium (EOM) processing supported under MPX-32. EOF only occurs when the beginning of the record to be transferred meets the specified condition. If EOF occurs in other than the first block of the transfer, the transfer is truncated up to the EOF block with the actual transfer count set to the truncated amount and no EOF indication. On the next sequential transfer after the truncated record, the task receives the EOF indication.

5.7.6 Software End-of-File for Unblocked Files

For unblocked files, the EOF indicator is the data pattern X'0FE0FE0F' in the first word of a disk sector. MPX-32 provides the M.WEOF service for nonbase tasks, or M_WRITEEOF service for base mode tasks, to generate the EOF indicator within a file. In this way, MPX-32 provides the capability to create multiple file files.

Use of multiple file unblocked files created with EOFM=T/Y (True or Yes respectively).

The EOF data pattern is not written or detected in unblocked files created with EOFM=T/Y. The M.WEOF service in nonbase assembler and the M_WRITEEOF service in base mode assembler, do not write an EOF pattern into an unblocked file created with EOFM=T/Y.

The EOF block recorded in the file descriptor for a file of this type is one block higher than the last sequentially written block of the file. MPX-32 does not expect the EOF block as the last block of the file.

The EOF pattern is detected on a read of an unblocked file, created with EOFM=T/Y. Any inconsistency between the EOF block recorded in the file descriptor and the last block of sequentially written data in the file (EOF block + 1) is corrected. The disk block containing the EOF data pattern is not part of the file contents.

Notes:

Use caution when accessing unblocked files in append mode that were created under a pre-3.4U02 revision of MPX-32, with EOFM=T/Y. Writing to such a file that has the EOF data pattern in the end-of-file block causes loss of data on a subsequent read of the EOF data sector.

Use the Media utility COPY command as described in the following procedure to avoid the problem described above:

- Create a scratch file for output.
- Use the MEDIA utility COPY command:

```
TSM>AS IN TO filename BLO=N
TSM>AS OT TO scratchfile BLO=N
TSM>MEDIA
MED>COPY,IN,OT
MED>EXIT
MED>END
```

- Delete the scratch file.

Existing multiple file unblocked files created with EOFM=T/Y can be preserved using VOLMGR COPY command, with options for REPLACE, and EOFM=F/N.

Table 5-6
EOF and EOM Description

		EOF	EOM
Blocked	Read	EOF occurs when the record with the EOF bit set in the block buffer control word is read.	EOM occurs when a record is read from a block that is past the last block written to the file.
	Write	Not applicable	EOM occurs when a record is written to a block that is one or more blocks past the end of the file space for a file that cannot be extended.
Unblocked With Data Formatting	Read	EOF occurs if the first word of data read has the data pattern of X'0FE0FE0F' in it. EOF will also occur if a read starts 1 block past the last block written to the file (EOF block in the FAT).	EOM occurs when a read starts from two or more blocks past the last block written to the file
	Write	Not applicable	Same as Blocked Write EOM
Unblocked With Data Formatting Inhibited	Read	EOF occurs if a read starts 1 block past the last block written to the file (EOF block in the FAT).	Same as Unblocked Read with Data Formatting EOM
	Write	Not applicable	Same as Blocked Write EOM

5.8 Spooled Output with Print or Punch Attribute

Real-time (not job related) permanent and temporary files are assigned with either the print or punch attribute. When the file is deassigned, spooled output is automatically generated on the auto-selectable print or punch device. If the file is temporary, it is deleted when the output process is complete. If the file is permanent, it is not deleted when the output process is complete.

5.9 Setting Up File Control Blocks for Device-Independent I/O

The MPX-32 Logical Device-independent I/O section in this chapter describes the function of an FCB. Parts of the FCB are required and must be defined by the user:

- a logical file code
- a transfer count and data buffer address for I/O operations controlled by this FCB

IOCS assumes the following if no other special I/O characteristics are defined in the FCB:

- wait I/O — IOCS returns to the calling task only when a requested operation on the file or device assigned to this FCB is complete
- automatic retry on error by IOCS and some handlers, as described in the Device Independent I/O Processing section in this chapter.
- device-dependent output and input are handled using standard techniques
- status information is returned in the FCB
- file and device access is sequential

Areas of the FCB define:

- no-wait I/O — Immediate return to the calling task after I/O operation is queued. The user can define the return address to return to in the task when processing is complete (normal or error).
- error processing inhibit — Only status is returned by handlers. No error processing is performed by IOCS or handlers.
- special device output characteristics
- random access for disk files

Some areas of the FCB are defined by IOCS. IOCS stores the opcode each time the task specifies a particular FCB, stores status returned by handlers, tracks actual record length in bytes for each transfer, and builds and maintains I/O queue and file assignment table (FAT) addresses. Refer to Appendix L for the 16-word format for the FCB.

5.9.1 Macros (M.DFCB/M.DFCBE)

The M.DFCB macro defines an 8-word FCB. The M.DFCBE macro defines a 16-word FCB.

There is no base mode equivalent service for M.DFCB. The base mode equivalent service for M.DFCBE is M_CREATEFCB.

Syntax

M.DFCB[E] *label, lfc, count, buffer,* [*error*] , [*random*] , [NWT] , [NER] , [DFI] , [NST] , [RAN] , [ASC I BIN] , [LDR I NLD] , [INT I PCK] , [EVN I BODD] , [556 I 800] , [*nowait*], [*nowaiterror*],[*buffaddr*]

label	contains an ASCII string to use as symbolic label for the address of this FCB.
lfc	contains a logical file code.
count	contains the transfer count. Specify in number of bytes.
buffer	contains the start address of the data buffer.
[*error*]	contains the error return address for wait I/O.
[*random*]	contains the random access address.
[NWT] , [NER] , [DFI] , [NST] , [RAN]	specifies general control flags. Refer to the FCB in Appendix L for flag descriptions.
[ASC I BIN]	for CR or CP
[LDR I NLD]	for PT (reader)
[INT I PCK]	for 7-track mag tape
[EVN I ODD]	for parity, s
[556 I 800]	for bits per inch density
[*nowait*]	specifies address for normal no-wait I/O end action return.
[*nowaiterror*]	specifies address for no-wait I/O error end action return.
[*buffaddr*]	specifies the address of a 192W user-supplied blocking buffer to be used for blocked I/O.

5.9.2 Sample FCB Set-up Nonmacro

User defines an FCB for terminal message output:

```
TERM        GEN       8/0,24/C'UT'
            GEN       12/B(TY.LEN),20/B(MESSAGE)
            REZ       6W
MESSAGE     EQU       $
            DATAB     C' "M"J CREATE FAILED.  ERRTYPE '
TY.LEN      EQU       $-MESSAGE
ERRTYPE     EQU       $-1B
```

Notes:

The source code uses `TERM` with M.WRIT to access this FCB. The logical file code is `UT`. The transfer control word built with the `GEN` directive has a transfer count equal to the message (`TY.LEN EQU $-MESSAGE`) computed by the Assembler.

The buffer start address is at `MESSAGE` (address is supplied by the Assembler). In the actual message, `"M` indicates carriage return and `"J` indicates line feed before output. The last byte of the message comes from R5 when an error occurs as defined by:

```
C.MSG   EQU         $
        M.CONBAD
        STB         R7,ERRTYPE
        M.WRIT      TERM
```

5.9.3 Sample FCB Set-up Macro

```
            M.DFCB    TERM, UT, TY.LEN, MESSAGE
MESSAGE     EQU       $
            DATAB     C' "M"J CREATE FAILED.  ERRTYPE '
TY.LEN      EQU       $-MESSAGE
ERRTYPE     EQU       $-1B
```

5.10 Setting Up TCPBs for the System Console

Messages are sent from a task to the system console and a response is optionally read back by a TCPB. The TCPB sets up task buffer areas for messages output by the task and reads back from the console.

The TCPB is comprised of a write and an optional read transfer control word defined like the TCW in word one of the FCB. If read back from the terminal is not desired, the read TCW must be zero. The user must perform his own carriage return and line feed.

The size of the read buffer should include space for both an input character count preceding the message (provided by IOCS) and the carriage return (end-of-record) character typed by the user at the end of an input line.

The count of input characters actually typed is placed by IOCS in the first byte of the read buffer. This input count does not include the carriage return character.

The byte transfer count is normally used by a task as maximum allowed input before termination. If the operator types in the maximum count without having typed a carriage return, the read is terminated.

The end-of-record character (carriage return) is normally allowed for in both the read and write buffer transfer count. If five characters are expected, the read transfer count would typically be for six characters. This allows the operator to type in and verify the accuracy of all five characters before terminating the input message by typing a carriage return.

Message transfers are always in bytes, so the buffer address must be a byte address (F-bit setting - bit 12).

If the NWT bit is set (word 2, bit 0), IOCS returns immediately to the calling task after the message is queued. The task can subsequently examine the OP indicator set by IOCS in word 2, bit 31 to see if the requested transfer is complete (0) or in process (1).

Table 5-7
Type Control Parameter Block

Type Control Parameter Block (TCPB) 24-bit Address

This is the preferred TCPB format. The M.TYPE and M_TYPE system service macros generate this format.

	0 1 7	8 31
Word 0	Output quantity (TCP.OQ)	Output data buffer address (TCP.OTCW)
1	Input quantity (TCP.IQ)	Input data buffer address (TCP.ITCW)
2	0 | 1 | Console device flags (TCP.FLGS)	

Bit interpretations for TCP.FLGS:

Bit	Meaning if set
0	no-wait I/O
1	data buffer addresses are 24-bit addresses (TCP.LAD) **Note:** This bit must be set.
31	operation in progress. This bit is reset after post-I/O processing completes.

Type Control Parameter Block (TCPB) 19-bit Address Compatible Mode

	0 11	12	13 31
Word 0	Output quantity (TCP.OQ)	1	Output data buffer address (TCP.OTCW)
1	Input quantity (TCP.IQ)	1	Input data buffer address (TCP.ITCW)
2	Console device flags (TCP.FLGS)		

Bit interpretations for TCP.FLGS:

Bit	Meaning if set
0	no-wait I/O
31	operation in progress. This bit is reset after post-I/O processing completes.

Note: Bit 12 of word 0 must be set.

5.11 MPX-32 Device-Dependent I/O

MPX-32 allows the user to perform direct channel I/O to F-class I/O devices. Direct channel I/O is accomplished by issuing a request to execute a logical or physical channel program that was built by the user.

A logical channel program allows the user to use the specific physical attributes of the device without having to be privileged or work with physical addressing. The user can assign a logical file code to a physical device, but must be concerned with the physical characteristics of the device. This gives the unprivileged user a way to use the explicit functionality provided by the device that may not be provided by device-independent I/O.

A physical channel program allows a privileged user to issue an I/O command list (IOCL) directly to a channel/controller/device. The IOCL is issued to the channel without any modification by the system. This allows a time critical user to issue physical I/O command lists that do not require the normal overhead of logical I/O.

5.11.1 Device-Dependent I/O Processing Overview

A task can issue device-dependent I/O by issuing an Execute Channel Program (EXCPM) request to IOCS for any XIO device allocated in the channel program access mode. This enables the task to pass an I/O channel program that it has built to a specified device with minimal IOCS overhead.

EXCPM support provides the user with the ability to perform direct channel I/O by using either a logical channel program or a physical channel program. A logical channel program is an IOCL built using logical addresses inside the building task's logical address space. The operating system will change all addresses to physical and resolve all physical address discontinuities by the use of data chaining before the IOCL is executed by the device. On completion of the I/O, control is returned to the task in the same manner as any I/O request.

A physical channel program is an IOCL ready to be executed by the device and is pointed to by a logical address. The area in memory containing the physical IOCL must be made unswappable by the user. All addresses in a physical channel program are physical addresses and discontinuities in the logical to physical mapping have been resolved by the user. A task must be privileged to execute a physical channel program.

A physical channel program started in the no-wait mode may have a post program-controlled interrupt (PPCI) EA receiver associated with it. This special EA receiver may be specified along with the normal EA receiver for no-wait I/O.

To issue an EXCPM to an 8-LAS device, that device must be in dual channel mode (i.e. FULL and NOECHO in the LOGONFLE, terminal set to HALF DUPLEX, and TSM OPTION UNQUIET in effect).

The EXCPM support applies only to extended I/O F-class devices.

The console does not have channel program capabilities.

5.11.2 Operational Description of Execute Channel Program (EXCPM)

The task must specify the logical address and a time-out value for the channel program in the FCB. The time-out value must be specified in seconds and placed in the second halfword of word 2 in the FCB. If the value equals zero, not time out is set for the EXCPM request. The logical address of the channel program must be placed in word 8 of the FCB. The task starts the I/O operation by issuing an SVC 1,X'25' or an M.CALL H.IOCS,10.

A SENSE buffer can be specified with an EXCPM request by placing the size of the buffer, in bytes, in FCB.SSIZ and the address of the buffer in FCB.SENA. If a SENSE buffer is specified, when the channel program terminates (normally or abnormally), a SENSE command is issued to the device and the information received is placed in the buffer specified. The size of the sense buffer must be equivalent to the number of bytes of sense status that a particular device will return. If the buffer size is zero, no sense information is returned and a buffer is not needed.

5.11.2.1 Logical Channel Program

The task builds a channel program inside its logical address space using the standard XIO commands and logical addressing. A logical IOCL must be contiguous in logical memory and TIC commands cannot be used to create loops.

5.11.2.2 Physical Channel Program

The task must be privileged to execute a physical channel program. It must build a physical IOCL using standard XIO commands and all related addresses must be physical. It must also resolve all data area discontinuities with the use of data chaining. After the physical IOCL is built, the task places the logical address of the IOCL into word 8 of the expanded FCB and sets bit 2 in word 2 of the FCB to indicate this is a physical IOCL. The user must insure that the memory containing the physical IOCL is unswappable.

5.11.2.3 Post Program-Controlled Interrupt (PPCI) End-Action Receiver

A privileged task may specify that a PPCI end-action receiver be used with a physical channel program issued in no-wait mode. This PPCI receiver will be entered when a PPCI occurs during the execution of the IOCL by the device. The logical address of the PPCI receiver must be placed in word 15 of the expanded FCB and bit 8 of word 2 of the FCB must be set.

The system builds a caller notification packet (CNP) to transfer control to the task's PPCI receiver. The address of the CNP is passed to the task's PPCI receiver in register three. The CNP contains pointers to a status queue where PPCI status is posted.

The status queue will be located immediately after the notification packet in memory. The default size of the status buffer is four doublewords. The user can specify the number of doublewords desired in the status buffer in the first byte of word 15 in the FCB.

When a PPCI is received by the handler's SI entry point, bit 8 of IOQ.CONT is checked to see if a PPCI receiver is present. If a PPCI receiver is not present, the interrupt level is exited without any status being posted. If a PPCI receiver is present, the status received from the PPCI is posted in the status buffer that follows the notification packet. When status is posted into the queue, the total PPCI received count (NOT.STAR) is incremented by one and the address of the next status location to use (NOT.STPT) is incremented by two words. If NOT.STPT points past the end of the status buffer, NOT.STPT is reset to point to the start of the buffer.

After status is posted, the handler checks to see if the notification packet is on the task interrupt queue in the task's DQE. If it is, the SI level is exited without issuing an end-action request. If the notification packet is not on the task interrupt queue, a request for end-action processing is made and the SI level is exited. If a task is at end-action level, it is possible that PPCI interrupts can occur which will cause status to be posted and another end-action request to be issued.

It is the task's responsibility to keep track of the number of PPCIs previously processed by the end-action routine. The task must also keep track of the position in the status buffer of the last status that was previously processed. This information will allow the end-action routine to know where the present status starts and whether any status was overwritten in the buffer (status was overwritten if the interrupts received minus the interrupts processed are greater than the length of the buffer).

The task uses the no-wait end-action return (SVC 1,X'2C') to exit the PPCI end-action level in the same manner that any no-wait end-action routine is exited.

5.11.2.4 Restrictions

The task must be privileged to execute a physical channel program and it must execute a physical channel program in no-wait mode to be able to have a PPCI end-action receiver.

The use of the TIC command to create loops is prohibited for logical channel programs. The use of the TESTSTAR and READ BACKWARD commands are also prohibited in logical channel programs.

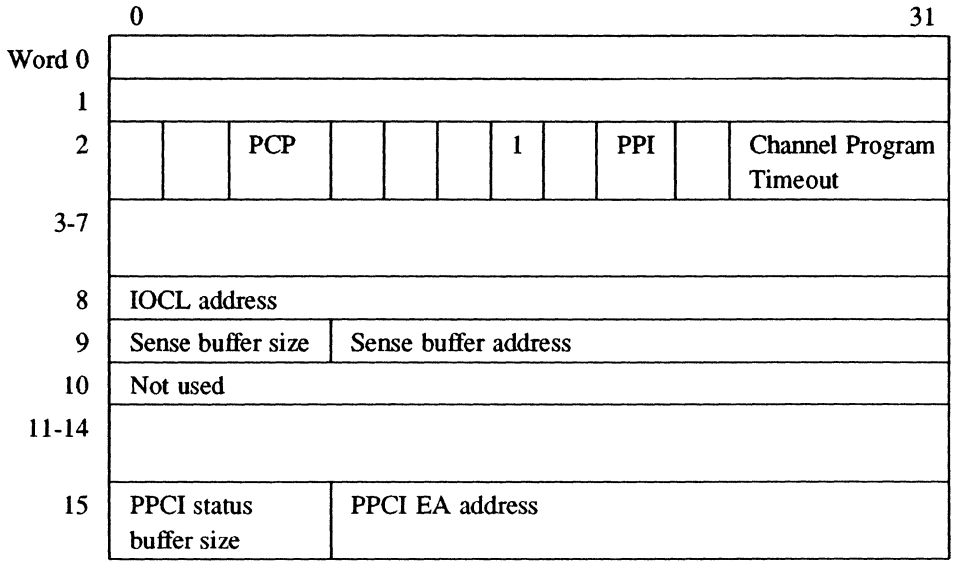
If the task does not want device I/O between channel programs, it must reserve the device exclusively while it is using it. This can be done by assigning the device for exclusive use or requesting an exclusive resource lock after the device is allocated. With this method there is no way to reserve a controller or channel.

If the task wants SENSE information to be returned on an error condition, it must set up a SENSE buffer to store the information. The SENSE buffer address must be placed in bits 8 - 31 of word 9 of the expanded FCB being used for the I/O. The length of the SENSE buffer, in bytes, must be placed in bits 0 - 7 of word 9 of the expanded FCB. If SENSE information is not wanted on error, word 9 of the FCB must be zero.

5.11.2.5 Setting Up File Control Blocks for EXCPM Requests

The differences between an FCB used for normal device-independent I/O and an FCB used for an EXCPM request are shown in the following table. The EXCPM FCB must be an expanded FCB.

Table 5-8
Execute Channel Program FCB Format



Word 2:

Bit	Description
2	contains the Physical Channel Program (PCP) if set. If not set, contains the Logical Channel Program.
8	contains the PPCI end-action receiver if set. If not set, there is no PPCI end-action receiver.

Word 8 contains the address of channel program to be executed

Word 9:

Bits	Description
0-7	contain the size of user-supplied sense buffer
8-31	contain the address of user-supplied sense buffer

Word 15:

Bits	Description
0-7	contain the size of the PPCI status buffer
8-31	contain the address of the PPCI end-action receiver

5.11.2.6 Post Program-Controlled Interrupt Notification Packet

If a task sets up a PPCI end-action receiver to check status during execution of its channel program, the status is returned in a notification packet. The address of the notification packet is contained in register three upon entering the task's PPCI end-action receiver. The notification packet is described in the following table.

Table 5-9
Notification Packet Layout for PPCI Receiver

	0 7	8 15	16 23	24 31
Word 0	String forward address (NOT.SFA)			
1	String backward address (NOT.SBA)			
2	Link priority (NOT.PRI)	NOT.TYPE See Note 1	Reserved	
3	FCB address (NOT.CODE)			
4	PSD 1 of task's PPCI receiver (NOT.PSD1)			
5	PSD 2 of task's PPCI receiver (NOT.PSD2)			
6	Number of PPCIs received since last buffer clear (NOT.STAR)		Number of status doublewords in status buffer (NOT.STAS)	
7	Address of PPCI status buffer (NOT.STAA)			
8	Address of buffer storing next status doubleword (NOT.STPT)			
9	Reserved			
10-*n*	PPCI status buffer			

Notes:

1. NOT.TYPE - Set to 1 for asynchronous notification.
2. Words 0-9 are updated by the operating system and must not be changed by the user.

5.11.2.7 Macros (M.FCBEXP)

The M.FCBEXP macro can define a file control block (FCB) for an Execute Channel Program request.

Syntax

M.FCBEXP *label,lfc* [, [*cpaddr*], [*tout*], [PCP], [NWI], [NST], [*ssize*], [*sbuffer*], [*nowait*], [*nowaiterror*], [*waiterror*], [*psize*] [,*ppciaddr*]]

label	is an ASCII string to use as symbolic label for address of FCB
lfc	is the logical file code; word 0, bits 8 - 31 of the FCB
[*cpaddr*]	is the logical address of channel program to be executed
[*tout*]	is the time-out value specified in seconds
[PCP]	specifies physical channel program
[NWI]	specifies no-wait I/O request
[NST]	is no status checking. All I/O appears to complete without error.
[*ssize*]	is the size of user-specified sense buffer
[*sbuffer*]	is the address of user-specified sense buffer
[*nowait*]	is the normal no-wait end-action return address
[*nowaiterror*]	is the no-wait end-action error return address
[*waiterror*]	is the wait I/O end-action error return address
[*psize*]	is the size of PPCI status buffer to use
[,*ppciaddr*]	is the PPCI end-action address

5.12 Resident Executive Services (H.REXS)

The H.REXS module replaces H.MONS. It provides interfaces for all major system services.

The following seven H.MONS entry points do not have equivalent H.REXS entry points and, therefore, are still valid by their H.MONS calls.

SVC	Macro	Entry Point	Description
1,X'40'	M.ALOC	H.MONS,21	Allocate file or peripheral device
1,X'41'	M.DALC	H.MONS,22	Deallocate file or peripheral device
1,X'42'	M.PDEV	H.MONS,1	Physical device inquiry
		H.MONS,2	Reserved
1,X'61'	M.CDJS	H.MONS,27	Submit job from disk file
1,X'73'	M.LOG	H.MONS,33	Permanent file log
1,X'74'	M.USER	H.MONS,34	User name specification

These entry points are included for compatibility purposes only and are described in the Compatible Services section of Chapter 6. The H.REXS services run faster than these H.MONS services and support more capabilities available in MPX-32.

All other H.REXS entry points perform the equivalent function as their H.MONS counterparts transparent to the user. See the Macro-Callable System Services section of Chapter 6 for H.REXS system service descriptions.

5.13 Resource Management (H.REMM)

The H.REMM module replaces H.ALOC. It performs the allocation and assignment of all system resources and is responsible for maintaining proper access compatibility and usage rights for these resources, such as files, directories, partitions, devices, and memory. The mechanisms for coordinating concurrent access to shared resources are also contained within this module.

Whenever H.REMM services are called by their macro names, any optional parameter not specified in the call is handled in one of the following ways:

- The appropriate register is assumed to have been previously loaded.
- The appropriate register is zeroed.

See the calling sequence description of each service to determine applicability of missing parameter handling.

The following five H.ALOC entry points do not have equivalent H.REMM entry points and, therefore, are still valid by their H.ALOC calls:

SVC	Macro	Entry Point	Description
1,X'1F'	M.SMULK	H.ALOC,19	unlock and dequeue shared memory
1,X'71'	M.SHARE	H.ALOC,12	share memory with another task
1,X'72'	M.INCL	H.ALOC,13	get shared memory (include)
1,X'79'	M.EXCL	H.ALOC,14	free shared memory (exclude)
N/A	N/A	H.ALOC,17	allocate file by space definition

These entry points are included for compatibility purposes only and are described in the Compatible Services section of Chapter 6. The H.REMM services run faster than these H.ALOC services and support more capabilities available in MPX-32.

All other H.REMM entry points perform the equivalent function as their H.ALOC counterparts transparent to the user. See the Macro-Callable System Services section of Chapter 6 for H.REMM system service descriptions.

5.14 Volume Management Module (H.VOMM)

The Volume Management Module (H.VOMM) manipulates the MPX-32 volume resident and related memory resident data structures to allow for creation, deletion, and maintenance of user and system resources residing on MPX-32 volumes.

Resources that reside on MPX-32 volumes include temporary files, directories, and the permanent files and memory partition definitions associated with the directories.

The volume resident data structures manipulated by H.VOMM include resource descriptors (RDs), the resource descriptor allocation map (DMAP), the file space allocation map (SMAP), the volume's directories, and their directory entries. The memory resident data structures include the mounted volume table (MVT) and the file assignment table (FAT).

5.14.1 H.VOMM Conventions

5.14.1.1 Entry Point Conventions

Entry points 1 through 16 and 23 are provided as user level services and are accessible through unprivileged SVC calls.

Entry points 17 through 22, 24, and 25 are meant for MPX-32 system usage, including internal H.VOMM usage, and are accessible only to privileged callers with M.CALL.

5.14.1.2 Pathnames

Pathnames uniquely identify a volume resident resource by explicitly or implicitly describing the volume, one or more directories, and the resource name. Pathnames consist of variable length ASCII character strings that are resolved on-line by H.VOMM in order to locate a resource.

H.VOMM supports a two level directory structure per volume including the root directory (level 1), and user directories (level 2). In the examples which follow, `VOL` means 1- to 16-character volume name, `DIR` means a 1- to 16-character directory name, and `FILE` means a 1- to 16-character resource name.

Example 1

```
@VOL(DIR)FILE
```

The `@VOL` component defines the volume where the resource `FILE` is located and implies that directory `DIR` should be located by the named volume's root directory. Furthermore, resource `FILE` should be located by directory `DIR`.

Example 2

```
FILE
```

The missing volume and directory components imply that resource FILE should be located by the user's current working volume and directory. The user's default current working volume and directory are established when the user enters the system and can be changed by the user.

Example 3

```
^(DIR)FILE
```

The ^ implies that the directory DIR should be located by the root directory of the user's current working volume, and that resource FILE should be located by user directory DIR.

Example 4

```
@SYSTEM(SYSTEM)FILE
```

The volume name SYSTEM and directory name SYSTEM are reserved names that identify the current system (IPL) volume and the special system directory on that volume.

5.14.1.3 Pathname Blocks (PNB)

The pathname block (PNB) is an alternative form of a pathname that can be used interchangeably with pathnames. Because of its structure, it can be parsed faster than a pathname. The PNB is a doubleword bounded, variable length ASCII character string which H.VOMM can distinguish from a pathname since the PNB always starts with an exclamation point.

H.VOMM provides a service to convert a pathname to a PNB. The examples which follow illustrate common pathnames and their corresponding PNB.

Example 1

```
@VOL1(DIR1)FILE1
```

Word 0	! V O L
1	blank
2	V O L 1
3	blank
4	blank
5	blank
6	! D I R
7	R O O T
8	D I R 1
9	blank
10	blank
11	blank
12	! R E S
13	blank
14	F I L E
15	1 ⌀ ⌀ ⌀
16	blank
17	blank

Example 2

```
FILE1
```

Word 0	! V O L
1	W O R K
2	! D I R
3	W O R K
4	! R E S
5	blank
6	F I L E
7	1 ⌀ ⌀ ⌀
8	blank
9	blank

Example 3

(DIRECTORY)MYFILE

Word 0	! V O L
1	W O R K
2	! D I R
3	R O O T
4	D I R E
5	C T O R
6	Y ∅ ∅ ∅
7	blank
8	! R E S
9	blank
10	M Y F I
11	L E ∅ ∅
12	blank
13	blank

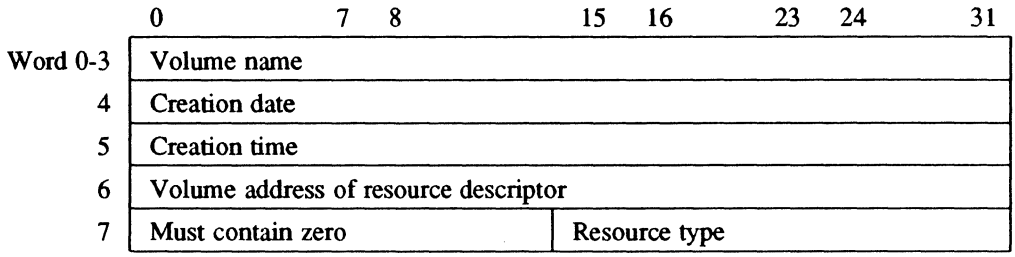
Example 4

@SYSTEM(SYSTEM)LOADMOD

Word 0	! V O L
1	S Y S T
2	! D I R
3	S Y S T
4	! R E S
5	blank
6	L O A D
7	M O D ∅
8	blank
9	blank

5.14.1.4 Resource Identifiers

The fastest means of locating a volume resource (once created) is by its resource identifier (must be on a doubleword boundary). The resource identifier has the following format:



Since the resource identifier contains the volume address of the resource descriptor, the resource descriptor (which points to and describes the resource) can be accessed directly without going through the various directories which would otherwise have to be traversed.

Given a valid pathname defining a resource, the corresponding resource descriptor may be retrieved by the H.VOMM locate resource service. The first eight words of a resource descriptor consist of the resource identifier.

5.14.1.5 Allocation Units

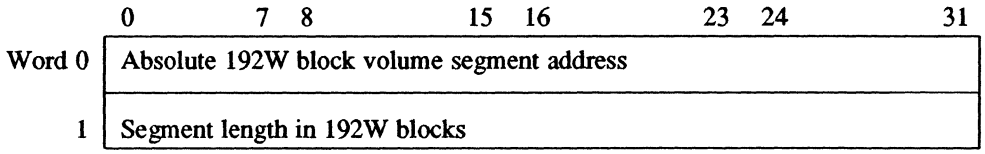
File space is allocated in allocation units. Allocation units are integral multiples of 192-word disk blocks. Allocation unit size is a volume parameter which is specified when the volume is formatted.

User calls to unprivileged H.VOMM entry points that allocate file space expect the amount of file space to be specified in terms of 192-word disk blocks. H.VOMM then rounds the number of blocks requested up to the nearest number of allocation units.

5.14.1.6 File Segment Definitions

The space associated with a file consists of one or more file segments. A file segment is a set of contiguous allocation units on a volume. When a file is created, its space consists of one or more file segments. As the file is expanded, new file segments become associated with the file. Different segments within one file are almost always discontiguous.

A file segment is defined by a 2 word file segment definition with the following format:



Up to 32 file segment definitions can be stored in one resource descriptor. A file cannot expand beyond 32 segments.

5.14.2 Calling/Return Parameter Conventions

5.14.2.1 Unused Register

No calling arguments are passed to an H.VOMM entry point by R3. Therefore, R3 is an unused register which is preserved across all H.VOMM entry point calls.

5.14.2.2 Specifying a Volume Resource

Whenever an H.VOMM entry point requires a volume resource to be identified (except H.VOMM,9) only one register is required regardless of whether pathname, pathname block, resource identifier, or logical file code (if the resource is assigned and opened) is supplied. Therefore, the following conventions are assumed by H.VOMM:

Identifier Supplied	Calling Register Format
Pathname Vector	0 7 8 31: Pathname length in bytes | Pathname address
Pathname Block Vector	0 7 8 31: Pathname block length in bytes | Pathname Block Address
Resource ID Vector	0 7 8 31: Resource ID length = 32 | Resource ID Address
Logical File Code	0 7 8 31: Must be zero | LFC
File Control Block	0 7 8 31: Must be zero | FCB Address

5.14.2.3 Status Codes

All H.VOMM entry points supply a status code to the caller indicating whether the entry point operation was successful and, if not, why not. A status code summary follows:

Status Code	Indication
0	operation successful
1	pathname invalid
2	pathname consists of volume only
3	volume not mounted
4	directory does not exist
5	directory name in use
6	directory creation not allowed at specified level
7	resource does not exist
8	resource name in use
9	resource descriptor unavailable
10	directory entry unavailable
11	required file space unavailable
12	unrecoverable I/O error while reading DMAP
13	unrecoverable I/O error while writing DMAP
14	unrecoverable I/O error while reading resource descriptor
15	unrecoverable I/O error while writing resource descriptor
16	unrecoverable I/O error while reading SMAP
17	unrecoverable I/O error while writing SMAP
18	unrecoverable I/O error while reading directory
19	unrecoverable I/O error while writing directory
20	project group name or key invalid
21	reserved
22	invalid FCB or LFC
23	parameter address specification error
24	resource descriptor not currently allocated
25	pathname block overflow
26	file space not currently allocated
27	change defaults not allowed
28	cannot access resource in requested mode or default system image file cannot be deleted
29	operation not allowed on this resource type
30	required parameter was not specified
31	file extension denied; segment definition area full
32	file extension denied; file would exceed maximum size allowed
33	I/O error occurred when resource was zeroed
34	replacement file cannot be allocated
35	invalid directory entry
36	directory and file not on same volume
37	an unimplemented entry point has been called
38	replacement file is not exclusively allocated to the caller
39	out of system space
40	cannot allocate FAT/FPT when creating a temporary file
41	deallocation error in zeroing file
42	resource descriptor destroyed or the resource descriptor and the directory entry linkage has been destroyed

Status Code	Indication
43	invalid resource specification
44	internal logic error from Resource Management Module (H.REMM). Abort task, try a different task. If it fails, reboot system.
45	attempted to modify more than one resource descriptor at the same time or attempted to rewrite resource descriptor prior to modifying it
46	unable to obtain resource descriptor lock (multiprocessor only)
47	directory contains active entries and cannot be deleted
48	a resource descriptor's link count is zero
49	attempting to delete a permanent resource without specifying a pathname or pathname block vector
50	resource descriptor contains unexpected resource descriptor type
51	directory entry deleted, but failed to release file space.
52	an attempt was made to deallocate free space or to allocate space that is currently allocated on a volume other than system disk
53	the file space create is less than the space requested
99	an attempt was made to deallocate free space or to allocate space that is currently allocated on the system volume

In some cases, H.VOMM also displays H.REMM abort conditions. If a user calls an H.VOMM service that calls an H.REMM service for processing and an abort condition occurs within the H.REMM processing, the abort condition is returned to H.VOMM. H.VOMM displays it to the user in the format 10*xx* where *xx* is the specific H.REMM abort condition. For example, abort condition 1026 indicates H.REMM error 26 has occurred. The TSM $ERR directive can determine the reason for the error; for example $ERR RM26.

5.14.2.4 Caller Notification Packet (CNP)

The CNP format is described in the H.REMM documentation. If a call to an H.VOMM entry point is accompanied by a CNP, the entry point status is always posted in the CNP. In addition, the caller can use the CNP to supply a denial return address that is to be taken if status is nonzero. If a CNP accompanies an H.VOMM call, status is nonzero, and no denial return address is supplied, a normal return is taken.

A CNP can be used to specify one or more options which are unique to the H.VOMM entry point called. To determine whether options apply, see the individual H.VOMM entry point descriptions.

If a CNP is not supplied, entry point status is always posted in register seven, a normal return is always taken, and no options may be specified.

In any case, whether a CNP is supplied or not, CC1 is always set if return status is nonzero.

5.14.2.5 Pathnames/Pathname Blocks

Whenever an H.VOMM entry point returns a nonzero status when it cannot completely resolve a pathname or pathname block, the address of the first unresolvable item within the pathname or pathname block is returned to the caller.

5.14.2.6 Resource Create Block (RCB)

Each H.VOMM entry point that creates a permanent file, a temporary file, a memory partition, or a directory may receive a resource create block (RCB) in order to fully define the attributes of the resource that is created. RCB formats are described in Tables 5-10, 5-11, and 5-12. RCBs must be doubleword bounded.

If an RCB is not supplied by the caller, the resource is created with the default attributes described in Chapter 4.

Table 5-10
Permanent and Temporary File Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0	File owner name (RCB.OWNR)
1	
2	File project group name (RCB.USER)
3	
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8	Maximum extension increment (RCB.MXEI). See Note 3.
9	Minimum extension increment (RCB.MNEI). See Note 4.
10	Maximum file size (RCB.MXSZ). See Note 5.
11	Original file size (RCB.OSIZ). See Note 6.
12	File starting address (RCB.ADDR). See Note 7.
13	File RID buffer (RCB.FAST). See Note 8.
14	Option flags (RCB.OPTS). See Note 9.
15	Default override (RCB.FREE). See Note 10.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
2	modify access allowed (RCB.MODI)
3	update access allowed (RCB.UPDA)
4	append access allowed (RCB.APPN)
9	delete access allowed (RCB.DELE)

2. Resource management flags. For any bit not set, system defaults apply and, in some cases, the default is the equivalent of the bit being set (optional):

Bit	Description
0-7	resource type, equivalent to file type code, interpreted as two hexadecimal digits, 0 - FF (RCB.FTYP)
8-10	reserved
11	file EOF management required (RCB.EOFM)
12	fast access (RCB.FSTF)
13	do not save (RCB.NSAV)
14	reserved for MPX-32 usage
15	file start block requested (RCB.SREQ)
16	file is executable (RCB.EXEC)
17	owner ID set on access (RCB.OWID)
18	project group ID set on access (RCB.UGID)
19	reserved
20	maximum file extension increment is zero. System default value not used. (RCB.MXEF)
21	minimum file extension increment is zero. System default value not used (RCB.MNEF)
22	reserved
23	zero file on creation/extension (RCB.ZERO)
24	file automatically extendible (RCB.AUTO)
25	file manually extendible (RCB.MANU)
26	file contiguity desired (RCB.CONT)
27	shareable (RCB.SHAR) (owner rights spec only)
28	link access (RCB.LINK)
29-30	reserved
31	file data initially recorded as blocked (RCB.BLOK)

3. Maximum extension increment is the desired file extension increment specified in blocks (optional). Default is 64 blocks.
4. Minimum extension increment is the minimum acceptable file extension increment specified in blocks (optional). Default is 32 blocks.
5. Maximum file size is the maximum extendible size for a file specified in blocks (optional).
6. Original file size is the original file size specified in blocks (optional). Default is 16 blocks.

7. File starting address is the disk block where the file should start, if possible. If the space needed is currently allocated, an error is returned (optional).
8. File RID buffer is the address within the file creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created file is not returned to the creating task.
9. Option flags bits are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)
3-6	reserved
7	multi-segment create
8	spool file type (RCB.SPOO)
9-15	reserved
16-23	maximum segment at creation (RCB.SEGN)
24-31	reserved

10. Default override - If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bit	Description
0-7	must be zero
8-10	reserved
11	file EOF management not required
12	fast access not required
13	resource can be saved
14-22	reserved
23	do not zero file on creation/extension
24	file is not automatically extendible
25	file is not manually extendible
26	file contiguity is not desired
27	resource is not shareable
28-30	reserved
31	file data initially recorded as unblocked

Table 5-11
Directory Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0-1	Directory owner name (RCB.OWNR)
2-3	Directory project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-10	Reserved
11	Directory original size (RCB.OSIZ). See Note 3.
12	Directory starting address (RCB.ADDR). See Note 4.
13	Directory RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications bits are as follows:

Bit	Description
0	read access allowed (RCB.READ)
8	directory may be traversed (RCB.TRAV)
9	directory may be deleted (RCB.DELE)
10	directory entries may be deleted (RCB.DEEN)
11	directory entries may be added (RCB.ADEN)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)
27	shareable (RCB.SHAR)

3. Directory original size is the number of entries required (optional).
4. Directory starting address is the disk block number where the directory should start, if possible. If the space needed is currently allocated, an error is returned (optional).
5. Directory RID buffer is the address within the directory creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created directory is not returned to the creating task.

6. Option flags are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)
3-31	reserved

7. If default override is set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional).

Bit	Description
0-7	must be zero
13	resource can be saved
27	resource is not shareable

Table 5-12
Nonbase Mode Memory Partition Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0-1	Partition owner name (RCB.OWNR)
2-3	Partition project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-9	Reserved
10	Starting word page number (RCB.PPAG)
11	Partition original size (RCB.OSIZ). See Note 3.
12	Partition starting address (RCB.ADDR). See Note 4.
13	Partition RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
9	delete access allowed (RCB.DELE)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)

3. Partition's original size is the number of protection granules required.
4. Partition's starting address is a 512-word protection granule number in the user's logical address space where the partition is to begin.
5. Partition's RID buffer is the address within the partition creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created partition is not returned to the creating task.
6. Option flags are optional:

Bits	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)
3-8	reserved
9	defines a static partition (RCB.STAT)
10-23	reserved
24-31	define memory class (RCB.MCLA). Values are:

Value	Memory Class
0	S (default)
1	E
2	H
3	S

7. If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bits	Description
0-7	must be zero
13	resource can be saved

5.14.3 Bad Block Handling

The process of initializing a volume with the media diagnostic and verification program produces information describing any sections of a disk that may not be used. This information may describe defective areas on the disk or areas that are reserved for use by the diagnostic hardware built into some disk drives.

This information is written to the disk and later read by J.VFMT and J.MOUNT. Using this information, these two programs mark all allocation units containing defective or reserved areas as allocated. This prevents H.VOMM from allocating space in a defective or reserved area.

5.14.4 Services

Whenever H.VOMM services are called by their macro names, any optional parameter not specified in the call is handled in one of the following ways:

- The appropriate register is assumed to have been previously loaded.
- The appropriate register is zeroed.

See the calling sequence description of each H.VOMM service in the Macro-Callable System Services section of Chapter 6 to determine applicability of missing parameter handling.

MPX-32 ™

System Services

Revision 3.5

Reference Manual Volume I(B)

April 1990

6 Nonbase Mode System Services
7 Base Mode System Services
A MPX-32 Device Access
B System Services Cross-Reference
C MPX-32 Abort and Crash Codes
D Numerical Information
E Powers of Integers
F ASCII Interchange Code Set
G IOP/MFP Panel Mode Commands
H Standard Date and Time Formats
I Compress Source Format
J Map Block Address Assignments
K Control Switches
L Data Structures
Glossary
Index

323-001551-600

Contents

Page

6 Nonbase Mode System Services

6.1 Overview 6-1
6.1.1 Syntax Rules and Descriptions 6-2
6.1.2 IPU Executable Nonbase Mode System Services 6-3
6.2 Macro-Callable System Services 6-4
6.2.1 M.ACTV - Activate Task 6-5
6.2.2 M.ADRS - Memory Address Inquiry 6-6
6.2.3 M.ANYW - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt 6-7
6.2.4 M.ASSN - Assign and Allocate Resource 6-8
6.2.5 M.ASYNCH - Set Asynchronous Task Interrupt 6-10
6.2.6 M.BACK - Backspace Record or File 6-11
6.2.7 M.BATCH - Batch Job Entry 6-13
6.2.8 M.BBTIM - Acquire Current Date/Time in Byte Binary Format 6-15
6.2.9 M.BORT - Abort Specified Task, Abort Self, or Abort with Extended Message 6-16
6.2.9.1 M.BORT - Specified Task 6-16
6.2.9.2 M.BORT - Self 6-17
6.2.9.3 M.BORT - With Extended Message 6-18
6.2.10 M.BRK - Break/Task Interrupt Link/Unlink 6-19
6.2.11 M.BRKXIT - Exit from Task Interrupt Level 6-19
6.2.12 M.BTIM - Acquire Current Date/Time in Binary Format 6-20
6.2.13 M.CLOSER - Close Resource 6-21
6.2.14 M.CLSE - Close File 6-23
6.2.15 M.CMD - Get Command Line 6-24
6.2.16 M.CONABB - Convert ASCII Date/Time to Byte Binary Format 6-25
6.2.17 M.CONADB - Convert ASCII Decimal to Binary 6-26
6.2.18 M.CONAHB - Convert ASCII Hexadecimal to Binary 6-27
6.2.19 M.CONASB - Convert ASCII Date/Time to Standard Binary 6-28
6.2.20 M.CONBAD - Convert Binary to ASCII Decimal 6-29
6.2.21 M.CONBAF - Convert Binary Date/Time to ASCII Format 6-30
6.2.22 M.CONBAH - Convert Binary to ASCII Hexadecimal 6-31
6.2.23 M.CONBBA - Convert Byte Binary Date/Time to ASCII 6-32

Contents

Page

6.2.24 M.CONBBY - Convert Binary Date/Time to Byte Binary 6-33
6.2.25 M.CONBYB - Convert Byte Binary Date/Time to Binary 6-34
6.2.26 M.CONN - Connect Task to Interrupt 6-35
6.2.27 M.CPERM - Create Permanent File.....6-37
6.2.28 M.CTIM - Convert System Date/Time Format 6-39
6.2.29 M.CWAT - System Console Wait 6-41
6.2.30 M.DASN - Deassign and Deallocate Resource.....6-42
6.2.31 M.DATE - Date and Time Inquiry.....6-44
6.2.32 M.DEBUG - Load and Execute Interactive Debugger 6-45
6.2.33 M.DEFT - Change Defaults 6-46
6.2.34 M.DELR - Delete Resource 6-48
6.2.35 M.DELTSK - Delete Task.....6-50
6.2.36 M.DEVID - Get Device Mnemonic or Type Code.....6-52
6.2.37 M.DIR - Create Directory.....6-53
6.2.38 M.DISCON - Disconnect Task from Interrupt 6-55
6.2.39 M.DLTT - Delete Timer Entry.....6-56
6.2.40 M.DMOUNT - Dismount Volume.....6-57
6.2.41 M.DSMI - Disable Message Task Interrupt.....6-59
6.2.42 M.DSUB - Disable User Break Interrupt.....6-60
6.2.43 M.DUMP - Memory Dump Request.....6-61
6.2.44 M.EAWAIT - End Action Wait 6-63
6.2.45 M.ENMI - Enable Message Task Interrupt.....6-64
6.2.46 M.ENUB - Enable User Break Interrupt.....6-65
6.2.47 M.ENVRMT - Get Task Environment.....6-66
6.2.48 M.EXCLUDE - Exclude Memory Partition.....6-67
6.2.49 M.EXIT - Terminate Task Execution 6-69
6.2.50 M.EXTD - Extend File.....6-70
6.2.51 M.FD - Free Dynamic Extended Indexed Data Space 6-72
6.2.52 M.FE - Free Dynamic Task Execution Space.....6-73
6.2.53 M.FWRD - Advance Record or File.....6-74
6.2.54 M.GADRL - Get Address Limits.....6-76
6.2.55 M.GADRL2 - Get Address Limits.....6-77
6.2.56 M.GD - Get Dynamic Extended Data Space 6-78
6.2.57 M.GDD - Get Dynamic Extended Discontiguous Data Space.....6-79
6.2.58 M.GE - Get Dynamic Task Execution Space 6-80
6.2.59 M.GETDEF - Get Definition for Terminal Function.....6-81
6.2.60 M.GMSGP - Get Message Parameters.....6-83
6.2.61 M.GRUNP - Get Run Parameters 6-84

Page

6.2.62 M.GTIM - Acquire System Date/Time in Any Format 6-85
6.2.63 M.GTSAD - Get TSA Start Address 6-86
6.2.64 M.HOLD - Program Hold Request 6-87
6.2.65 M.ID - Get Task Number 6-88
6.2.66 M.INCLUDE - Include Memory Partition 6-90
6.2.67 M.INQUIRY - Resource Inquiry 6-93
6.2.68 M.INT - Activate Task Interrupt 6-97
6.2.69 M.IPUBS - Set IPU Bias 6-98
6.2.70 M.LOC - Read Descriptor 6-99
6.2.71 M.LOCK - Set Exclusive Resource Lock 6-101
6.2.72 M.LOGR - Log Resource or Directory 6-103
6.2.72.1 Resource Specifications for Pathnames 6-103
6.2.72.2 Resource Specifications for Pathname Blocks 6-104
6.2.72.3 Resource Specifications for a Resource Identifier 6-104
6.2.72.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index 6-104
6.2.73 M.MEM - Create Memory Partition 6-108
6.2.74 M.MEMB - Get Memory in Byte Increments 6-110
6.2.75 M.MEMFRE - Free Memory in Byte Increments 6-111
6.2.76 M.MOD - Modify Descriptor 6-112
6.2.77 M.MODU - Modify Descriptor User Area 6-114
6.2.78 M.MOUNT - Mount Volume 6-115
6.2.79 M.MOVE - Move Data to User Address 6-117
6.2.80 M.MYID - Get Task Number 6-118
6.2.81 M.NEWRRS - Reformat RRS Entry 6-119
6.2.82 M.OLAY - Load Overlay Segment 6-121
6.2.83 M.OPENR - Open Resource 6-122
6.2.84 M.OSREAD - Physical Memory Read 6-124
6.2.85 M.OSWRIT - Physical Memory Write 6-125
6.2.86 M.PGOD - Task Option Doubleword Inquiry 6-126
6.2.87 M.PGOW - Task Option Word Inquiry 6-127
6.2.88 M.PNAM - Reconstruct Pathname 6-128
6.2.89 M.PNAMB - Convert Pathname to Pathname Block 6-129
6.2.90 M.PRIL - Change Priority Level 6-131
6.2.91 M.PRIV - Reinstate Privilege Mode to Privilege Task 6-132
6.2.92 M.PTSK - Parameter Task Activation 6-133
6.2.93 M.QATIM - Acquire Current Date/Time in ASCII Format 6-138
6.2.94 M.RADDR - Get Real Physical Address 6-139
6.2.95 M.RCVR - Receive Message Link Address 6-140

Contents

Page

6.2.96 M.READ - Read Record 6-141
6.2.97 M.RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode 6-142
6.2.98 M.RENAM - Rename File 6-143
6.2.99 M.REPLAC - Replace Permanent File 6-144
6.2.100 M.RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode 6-145
6.2.101 M.REWRIT - Rewrite Descriptor 6-146
6.2.102 M.REWRTU - Rewrite Descriptor User Area 6-147
6.2.103 M.ROPL - Reset Option Lower 6-148
6.2.104 M.RRES - Release Channel Reservation 6-149
6.2.105 M.RSML - Resourcemark Lock 6-150
6.2.106 M.RSMU - Resourcemark Unlock 6-152
6.2.107 M.RSRV - Reserve Channel 6-153
6.2.108 M.RWND - Rewind File 6-154
6.2.109 M.SETS - Set User Status Word 6-155
6.2.110 M.SETSYNC - Set Synchronous Resource Lock 6-157
6.2.111 M.SETT - Create Timer Entry 6-159
6.2.112 M.SMSGR - Send Message to Specified Task 6-162
6.2.113 M.SOPL - Set Option Lower 6-163
6.2.114 M.SRUNR - Send Run Request to Specified Task 6-164
6.2.115 M.SUAR - Set User Abort Receiver Address 6-166
6.2.116 M.SUME - Resume Task Execution 6-167
6.2.117 M.SURE - Suspend/Resume 6-168
6.2.118 M.SUSP - Suspend Task Execution 6-169
6.2.119 M.SYNCH - Set Synchronous Task Interrupt 6-170
6.2.120 M.TBRKON - Trap Online User's Task 6-171
6.2.121 M.TDAY - Time-of-Day Inquiry 6-172
6.2.122 M.TEMP - Create Temporary File 6-173
6.2.123 M.TEMPER - Change Temporary File to Permanent File 6-175
6.2.124 M.TRNC - Truncate File 6-177
6.2.125 M.TSCAN - Scan Terminal Input Buffer 6-178
6.2.126 M.TSMPC - TSM Procedure Call 6-179
6.2.127 M.TSTE - Arithmetic Exception Inquiry 6-182
6.2.128 M.TSTS - Test User Status Word 6-183
6.2.129 M.TSTT - Test Timer Entry 6-184
6.2.130 M.TURNON - Activate Program at Given Time-of-Day 6-185
6.2.131 M.TYPE - System Console Type 6-187
6.2.132 M.UNLOCK - Release Exclusive Resource Lock 6-188
6.2.133 M.UNSYNC - Release Synchronous Resource Lock 6-190

Page

6.2.134 M.UPRIV - Change Task to Unprivileged Mode 6-192
6.2.135 M.UPSP - Upspace 6-193
6.2.136 M.VADDR - Validate Address Range 6-194
6.2.137 M.WAIT - Wait I/O 6-195
6.2.138 M.WEOF - Write EOF 6-196
6.2.139 M.WRIT - Write Record 6-197
6.2.140 M.XBRKR - Exit from Task Interrupt Level 6-198
6.2.141 M.XIEA - No-Wait I/O End-Action Return 6-199
6.2.142 M.XMEA - Exit from Message End-Action Routine 6-200
6.2.143 M.XMSGR - Exit from Message Receiver 6-201
6.2.144 M.XREA - Exit from Run Request End-Action Routine 6-202
6.2.145 M.XRUNR - Exit Run Receiver 6-203
6.2.146 M.XTIME - Task CPU Execution Time 6-204
6.3 Nonmacro-Callable System Services 6-205
6.3.1 Allocate File Space 6-206
6.3.1.1 Clean-up Mode 6-206
6.3.1.2 Normal Mode 6-206
6.3.2 Allocate Resource Descriptor 6-208
6.3.3 Create Temporary File 6-209
6.3.3.1 VOMM Internal Call 6-209
6.3.3.2 External Call 6-209
6.3.3.3 Default File Attributes 6-209
6.3.3.4 Volume Selection 6-209
6.3.4 Deallocate File Space 6-211
6.3.5 Deallocate Resource Descriptor 6-212
6.3.6 Debug Link Service 6-213
6.3.7 Eject/Purge Routine 6-214
6.3.8 Erase or Punch Trailer 6-215
6.3.9 Execute Channel Program 6-216
6.3.10 Get Extended Memory Array 6-217
6.3.11 Read/Write Authorization File 6-218
6.3.12 Release FHD Port 6-219
6.3.13 Reserve FHD Port 6-220
6.4 Compatible System Services 6-221
6.4.1 M.ALOC - Allocate File or Peripheral Device 6-222
6.4.2 M.CDJS - Submit Job from Disk File 6-226
6.4.3 M.CREATE - Create Permanent File 6-228
6.4.4 M.DALC - Deallocate File or Peripheral Device 6-231

Page

6.4.5 M.DELETE - Delete Permanent File or Non-SYSGEN Memory Partition 6-232
6.4.6 M.EXCL - Free Shared Memory 6-233
6.4.7 M.FADD - Permanent File Address Inquiry 6-234
6.4.8 M.FILE - Open File 6-236
6.4.9 M.FSLR - Release Synchronization File Lock 6-237
6.4.10 M.FSLS - Set Synchronization File Lock 6-238
6.4.11 M.FXLR - Release Exclusive File Lock 6-240
6.4.12 M.FXLS - Set Exclusive File Lock 6-241
6.4.13 M.INCL - Get Shared Memory 6-242
6.4.14 M.LOG - Permanent File Log 6-244
6.4.15 M.PDEV - Physical Device Inquiry 6-246
6.4.16 M.PERM - Change Temporary File to Permanent 6-248
6.4.17 M.SHARE - Share Memory with Another Task 6-250
6.4.18 M.SMULK - Unlock and Dequeue Shared Memory 6-252
6.4.19 M.USER - User Name Specification 6-253

7 Base Mode System Services

7.1 General Description 7-1
7.1.1 Syntax Rules and Descriptions 7-2
7.1.1.1 Parameter Specification 7-2
7.1.2 IPU Executable Base Mode System Services 7-5
7.2 Macro-Callable System Services 7-6
7.2.1 M_ACTV - Activate Task 7-7
7.2.2 M_ADRS - Memory Address Inquiry 7-8
7.2.3 M_ADVANCE - Advance Record or File 7-9
7.2.4 M_ANYWAIT - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt 7-11
7.2.5 M_ASSIGN - Assign and Allocate Resource 7-12
7.2.6 M_ASYNCH - Set Asynchronous Task Interrupt 7-14
7.2.7 M_AWAITACTION - End Action Wait 7-15
7.2.8 M_BACKSPACE - Backspace Record or File 7-16
7.2.9 M_BATCH - Batch Job Entry 7-18
7.2.10 M_BBTIM - Acquire Current Date/Time in Byte Binary Format 7-20
7.2.11 M_BORT - Abort Specified Task, Abort Self, or Abort with Extended Message 7-21
7.2.11.1 M_BORT - Abort Specified Task 7-21
7.2.11.2 M_BORT - Abort Self 7-22

Page

7.2.11.3 M_BORT - Abort with Extended Message 7-23
7.2.12 M_BRK - Break/Task Interrupt Link/Unlink 7-24
7.2.13 M_BRKXIT - Exit from Task Interrupt Level 7-24
7.2.14 M_BTIM - Acquire Current Date/Time in Binary Format 7-25
7.2.15 M_CHANPROGFCB - Execute Channel Program File Control Block 7-26
7.2.16 M_CLOSER - Close Resource 7-27
7.2.17 M_CLSE - Close File 7-29
7.2.18 M_CMD - Get Command Line 7-30
7.2.19 M_CONABB - Convert ASCII Date/Time to Byte Binary Format 7-31
7.2.20 M_CONADB - Convert ASCII Decimal to Binary 7-32
7.2.21 M_CONAHB - Convert ASCII Hexadecimal to Binary 7-33
7.2.22 M_CONASB - Convert ASCII Date/Time to Standard Binary 7-34
7.2.23 M_CONBAD - Convert Binary to ASCII Decimal 7-35
7.2.24 M_CONBAF - Convert Binary Date/Time to ASCII Format ..7-36
7.2.25 M_CONBAH - Convert Binary to ASCII Hexadecimal 7-37
7.2.26 M_CONBBA - Convert Byte Binary Date/Time to ASCII.....7-38
7.2.27 M_CONBBY - Convert Binary Date/Time to Byte Binary 7-39
7.2.28 M_CONBYB - Convert Byte Binary Date/Time to Binary 7-40
7.2.29 M_CONN - Connect Task to Interrupt 7-41
7.2.30 M_CONSTRUCTPATH - Reconstruct Pathname 7-42
7.2.31 M_CONVERTTIME - Convert Time 7-43
7.2.32 M_CREATEFCB - Create File Control Block 7-45
7.2.33 M_CREATEP - Create Permanent File 7-46
7.2.34 M_CREATET - Create Temporary File.....7-48
7.2.35 M_CTIM - Convert System Date/Time Format 7-50
7.2.36 M_CWAT - System Console Wait 7-51
7.2.37 M_DATE - Date and Time Inquiry.....7-52
7.2.38 M_DEASSIGN - Deassign and Deallocate Resource 7-53
7.2.39 M_DEBUG - Load and Execute Interactive Debugger 7-55
7.2.40 M_DEFT - Change Defaults 7-56
7.2.41 M_DELETER - Delete Resource 7-57
7.2.42 M_DELTSK - Delete Task.....7-59
7.2.43 M_DEVID - Get Device Mnemonic or Type Code.....7-60
7.2.44 M_DIR - Create Directory.....7-61
7.2.45 M_DISCON - Disconnect Task from Interrupt 7-63
7.2.46 M_DISMOUNT - Dismount Volume 7-64
7.2.47 M_DLTT - Delete Timer Entry.....7-66

Contents

Page

7.2.48 M_DSMI - Disable Message Task Interrupt 7-67
7.2.49 M_DSUB - Disable User Break Interrupt 7-68
7.2.50 M_DUMP - Memory Dump Request 7-69
7.2.51 M_ENMI - Enable Message Task Interrupt 7-70
7.2.52 M_ENUB - Enable User Break Interrupt 7-71
7.2.53 M_ENVRMT - Get Task Environment 7-72
7.2.54 M_EXCLUDE - Exclude Shared Image 7-73
7.2.55 M_EXIT - Terminate Task Execution 7-75
7.2.56 M_EXTENDFILE - Extend File 7-76
7.2.57 M_EXTSTS - Exit With Status 7-78
7.2.58 M_FREEMEMBYTES - Free Memory in Byte Increments 7-79
7.2.59 M_GETCTX - Get User Context 7-80
7.2.60 M_GETDEF - Get Definition for Terminal Function 7-81
7.2.61 M_GETMEMBYTES - Get Memory in Byte Increments 7-83
7.2.62 M_GETTIME - Get Current Date and Time 7-84
7.2.63 M_GMSGP - Get Message Parameters 7-86
7.2.64 M_GRUNP - Get Run Parameters 7-87
7.2.65 M_GTIM - Acquire System Date/Time in Any Format 7-88
7.2.66 M_GTSAD - Get TSA Start Address 7-89
7.2.67 M_HOLD - Program Hold Request 7-90
7.2.68 M_ID - Get Task Number 7-91
7.2.69 M_INCLUDE - Include Shared Image 7-93
7.2.70 M_INQUIRER - Resource Inquiry 7-96
7.2.71 M_INT - Activate Task Interrupt 7-101
7.2.72 M_IPUBS - Set IPU Bias 7-102
7.2.73 M_LIMITS - Get Base Mode Task Address Limits 7-103
7.2.74 M_LOCK - Set Exclusive Resource Lock 7-104
7.2.75 M_LOGR - Log Resource or Directory 7-106
7.2.75.1 Resource Specifications for Pathnames 7-106
7.2.75.2 Resource Specifications for Pathname Blocks 7-107
7.2.75.3 Resource Specifications for a Resource Identifier 7-107
7.2.75.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index 7-107
7.2.76 M_MEM - Create Memory Partition 7-111
7.2.77 M_MOD - Modify Descriptor 7-113
7.2.78 M_MODU - Modify Descriptor User Area 7-115
7.2.79 M_MOUNT - Mount Volume 7-116
7.2.80 M_MOVE - Move Data to User Address 7-118
7.2.81 M_MYID - Get Task Number 7-120
7.2.82 M_OPENR - Open Resource 7-121

Page

7.2.83 M_OPTIONDWORD - Task Option Doubleword Inquiry 7-124
7.2.84 M_OPTIONWORD - Task Option Word Inquiry 7-125
7.2.85 M_OSREAD - Physical Memory Read 7-126
7.2.86 M_OSWRIT - Physical Memory Write 7-127
7.2.87 M_PNAMB - Convert Pathname to Pathname Block 7-129
7.2.88 M_PRIL - Change Priority Level.....7-131
7.2.89 M_PRIVMODE - Reinstate Privilege Mode to Privilege Task.....7-132
7.2.90 M_PTSK - Parameter Task Activation 7-133
7.2.91 M_PUTCTX - Put User Context.....7-138
7.2.92 M_QATIM - Acquire Current Date/Time in ASCII Format..7-139
7.2.93 M_RADDR - Get Real Physical Address 7-140
7.2.94 M_RCVR - Receive Message Link Address 7-141
7.2.95 M_READ - Read Record.....7-142
7.2.96 M_READD - Read Descriptor 7-144
7.2.97 M_RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode 7-145
7.2.98 M_RENAME - Rename File 7-146
7.2.99 M_REPLACE - Replace Permanent File 7-147
7.2.100 M_RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode 7-148
7.2.101 M_REWIND - Rewind File.....7-149
7.2.102 M_REWRIT - Rewrite Descriptor 7-150
7.2.103 M_REWRTU - Rewrite Descriptor User Area 7-151
7.2.104 M_ROPL - Reset Option Lower.....7-152
7.2.105 M_RRES - Release Channel Reservation 7-153
7.2.106 M_RSML - Resourcemark Lock 7-154
7.2.107 M_RSMU - Resourcemark Unlock 7-155
7.2.108 M_RSRV - Reserve Channel.....7-156
7.2.109 M_SETERA - Set Exception Return Address.....7-157
7.2.110 M_SETEXA - Set Exception Handler.....7-158
7.2.111 M_SETS - Set User Status Word 7-159
7.2.112 M_SETSYNC - Set Synchronous Resource Lock 7-161
7.2.113 M_SETT - Create Timer Entry 7-163
7.2.114 M_SMSGR - Send Message to Specified Task 7-166
7.2.115 M_SOPL - Set Option Lower.....7-167
7.2.116 M_SRUNR - Send Run Request to Specified Task.....7-168
7.2.117 M_SUAR - Set User Abort Receiver Address.....7-170
7.2.118 M_SUME - Resume Task Execution 7-171

Contents

Page

7.2.119 M_SURE - Suspend/Resume 7-172
7.2.120 M_SUSP - Suspend Task Execution 7-173
7.2.121 M_SYNCH - Set Synchronous Task Interrupt 7-174
7.2.122 M_TBRKON - Trap Online User's Task 7-175
7.2.123 M_TDAY - Time-of-Day Inquiry 7-176
7.2.124 M_TEMPFILETOPERM - Change Temporary File to Permanent File 7-177
7.2.125 M_TRUNCATE - Truncate File 7-179
7.2.126 M_TSCAN - Scan Terminal Input Buffer 7-180
7.2.127 M_TSMPC - TSM Procedure Call 7-181
7.2.128 M_TSTE - Arithmetic Exception Inquiry 7-184
7.2.129 M_TSTS - Test User Status Word 7-185
7.2.130 M_TSTT - Test Timer Entry 7-186
7.2.131 M_TURNON - Activate Program at Given Time-of-Day 7-187
7.2.132 M_TYPE - System Console Type 7-189
7.2.133 M_UNLOCK - Release Exclusive Resource Lock 7-190
7.2.134 M_UNPRIVMODE - Change Task to Unprivileged Mode 7-192
7.2.135 M_UNSYNC - Release Synchronous Resource Lock 7-193
7.2.136 M_UPSP - Upspace 7-195
7.2.137 M_VADDR - Validate Address Range 7-196
7.2.138 M_WAIT - Wait I/O 7-197
7.2.139 M_WRITE - Write Record 7-198
7.2.140 M_WRITEEOF - Write EOF 7-199
7.2.141 M_XBRKR - Exit from Task Interrupt Level 7-200
7.2.142 M_XIEA - No-Wait I/O End-Action Return 7-201
7.2.143 M_XMEA - Exit from Message End-Action Routine 7-202
7.2.144 M_XMSGR - Exit from Message Receiver 7-203
7.2.145 M_XREA - Exit from Run Request End-Action Routine 7-204
7.2.146 M_XRUNR - Exit Run Receiver 7-205
7.2.147 M_XTIME - Task CPU Execution Time 7-206
7.3 Nonmacro-Callable System Services 7-207
7.3.1 Debug Link Service 7-207
7.3.2 Eject/Purge Routine 7-208
7.3.3 Erase or Punch Trailer 7-209
7.3.4 Execute Channel Program 7-210
7.3.5 Get Extended Memory Array 7-211
7.3.6 Release FHD Port 7-212
7.3.7 Reserve FHD Port 7-212

Page

A MPX-32 Device Access A-1

B System Services Cross-Reference B-1

C MPX-32 Abort and Crash Codes C-1

D Numerical Information D-1

E Powers of Integers E-1

F ASCII Interchange Code Set F-1

G IOP/MFP Panel Mode Commands G-1

H Standard Date and Time Formats H-1

I Compressed Source Format I-1

J Map Block Address Assignments J-1

K Control Switches K-1

L Data Structures L-1

Glossary GL-1

Index IN-1

6 Nonbase Mode System Services

6.1 Overview

MPX-32 resident nonbase mode system service routines perform frequently required operations with maximum efficiency. Using the Supervisor Call instruction, tasks running in any environment can call these routines.

All system service routines are reentrant. Thus, each service routine is always available to the task that is currently active.

System service routines are provided as standard modular components of the MPX-32. The open-ended design of the system allows routines to be added to tailor MPX-32 to a specific application.

System services enable tasks to:

- activate, suspend, resume, abort, terminate, and hold task execution
- change a task's priority level
- create, test, and delete timers
- interrogate system clocks
- allocate and deallocate devices and files
- obtain the characteristics of a device or file
- communicate with other tasks with messages and status words
- load and execute overlays
- obtain information about the memory assigned to a task
- connect tasks to interrupts
- determine the arithmetic exception and option word status for a task

MPX-32 services are implemented as SVC traps. There are several ways of accessing services:

1. By macro calls, with parameter passing as indicated in the individual descriptions. The expansion code in the system macro library is then accessed automatically during assembly to provide assembly language setup of appropriate registers and instructions, including SVCs, in the user code.
2. By setting up appropriate registers and instructions directly and using appropriate SVCs.
3. By following number 2 above but issuing an M.CALL request to the entry point of the system module that provides the service.

The first two access paths are described for each system service. The third access path is privileged, and is indicated primarily to provide the appropriate system module names and entry point numbers for cross-reference to other documentation when needed.

Special operations performed for a task are:

- Open — If not issued by the task, IOCS opens the file or device for the default access in effect at that time.
- Close — If not issued by the task, the file is closed automatically and a device is deallocated automatically during task termination.

Callable system services are described in alphabetical order by macro name. Available system services that are not macro callable are described in the Nonmacro-Callable System Services section.

Services for interactive tasks are described under Job Control Language.

6.1.1 Syntax Rules and Descriptions

System services can be called by their macro name, their SVC number, or their module entry point number. It is recommended that whenever possible the macro name be used. When a macro name is used, any optional parameter not specified in the call is handled as follows:

- the appropriate register is assumed to have been previously loaded

  (or)

- the appropriate register will be zeroed

Refer to the Calling Sequence description of each service to determine applicability of missing parameter handling.

Defaults for optional parameters are documented in the description of each service.

When a required parameter is not specified or an invalid parameter is specified, an error message is displayed in the listing regardless of the listing controls in effect.

The integrity of the condition code setting on exit is not guaranteed for system services, except as documented for a particular service. Refer to the description of each service to determine whether the exit condition code settings are applicable.

6.1.2 IPU Executable Nonbase Mode System Services

Once a task has gained entry into the IPU, there is a limited set of system services that the IPU can execute. These are memory reference only system services, since the IPU can not execute any I/O instructions. The following nonbase mode system services are executable in the IPU:

SVC	Description
M.ADRS	Memory Address Inquiry
M.BBTIM	Acquire Current Date/Time in Byte Binary Format
M.BTIM	Acquire Current Date/Time in Binary Format
M.CMD	Get Command Line
M.CONABB	Convert ASCII Date/Time to Byte Binary Format
M.CONADB	Convert ASCII Decimal to Binary
M.CONAHB	Convert ASCII Hexadecimal to Binary
M.CONASB	Convert ASCII Date/Time to Standard Binary
M.CONBAD	Convert Binary to ASCII Decimal
M.CONBAF	Convert Binary Date/Time to ASCII Format
M.CONBAH	Convert Binary to ASCII Hexadecimal
M.CONBBA	Convert Byte Binary Date/Time to ASCII
M.CONBBY	Convert Binary Date/Time to Byte Binary
M.CONBYB	Convert Byte Binary Date/Time to Binary
M.CTIM	Convert System Date/Time Format
M.DATE	Date and Time Inquiry
M.DEVID	Get Device Mnemonic or Type
M.DSMI	Disable Message Task Interrupt
M.DSUB	Disable User Break Interrupt
M.ENUB	Enable User Break Interrupt
M.ENVRMT	Get Task Environment
M.GTIM	Acquire System Date and Time in any Format
M.GTSAD	Get TSA Start Address
M.OSREAD	Physical Memory Read
M.OSWRIT	Physical Memory Write
M.PGOD	Task Option Doubleword Inquiry
M.PGOW	Task Option Word Inquiry
M.QATIM	Acquire Current Date/Time in ASCII Format
M.SYNCH	Set Synchronous Task Interrupts
M.TDAY	Time-of-Day Inquiry
M.TSTE	Arithmetic Exception Inquiry
M.TSTT	Test Timer Entry

6.2 Macro-Callable System Services

All nonbase mode system services are described in detail in the pages which follow, arranged alphabetically by their macro name. System services which are supported for nonbase mode tasks are prefaced by the characters "M.".

6.2.1 M.ACTV - Activate Task

The M.ACTV service activates a task. The task assumes the owner name of the caller. When a load module is supplied as input, the operating system searches in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied as input.

The base mode equivalent service is M_ACTV.

Entry Conditions

Calling Sequence

M.ACTV *loadmod*

(or)

LD R6,*vector*
SVC 1,X'52' (or) M.CALL H.REXS,15

loadmod is a doubleword containing the 1- to 8-ASCII character name left-justified and blank filled, of a system load module for which an activation request is to be queued. R7 contains the vector value; R6 is zero.

vector the pathname vector or RID vector pointing to the load module to be activated. *vector* must be supplied if the load module to be activated is not in the system directory.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 contains zero if the service was performed
R7 contains the task activation number

(or)

R6 contains one if the task was already active
R7 task number of existing task with same name

(or)

R6	Value	Description
	2	load module file not in directory
	3	unable to allocate load module
	4	file is not a valid load module
	5	DQE is not available
	6	read error on resource descriptor
	7	read error on load module
	8	insufficient logical/physical address space for task activation
	40	invalid load module
	84	invalid logical position for extended MPX-32

R7 equals zero

6.2.2 M.ADRS - Memory Address Inquiry

The M.ADRS service provides the beginning and ending logical addresses of the memory allocated to a task. The beginning address is the location into which the first word was loaded and is a word address. The ending address is also a word address and defines the last word allocated to the task.

This service can be executed by the IPU.

The base mode equivalent service is M_ADRS.

Entry Conditions

Calling Sequence

M.ADRS

(or)

SVC 1,X'44' (or) M.CALL H.REXS,3

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6 logical word address of the first location of the task's DSECT. This address is always on a page boundary.

R7 logical word address of the last location available for loading or expansion of the task's DSECT. This address is always on a map block boundary minus one word.

6.2.3 M.ANYW - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt

The M.ANYW service places the currently executing task in a state waiting for the completion of any no-wait request, for the receipt of a message, or for a break interrupt. The task is removed from the associated ready-to-run list, and placed in the any-wait list. A return is made to the program location following the SVC instruction only when one of the wait conditions has been satisfied or when the optional time-out value has expired.

The base mode equivalent service is M_ANYWAIT.

Entry Conditions

Calling Sequence

M.ANYW *time1*

(or)

```
LW   R6,time1
SVC  1,X'7C'  (or)  M.CALL H.REXS,37
```

time1 contains zero if wait for an indefinite period is requested. Otherwise, *time1* contains the negative number of time units to elapse before the wait is terminated.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
MS31    USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN
        END-ACTION ROUTINE
```

6.2.4 M.ASSN - Assign and Allocate Resource

The M.ASSN service associates a resource with a logical file code (LFC) used by a process and allocates the resource. This function creates a FAT/FPT pair within the user's TSA and associates an allocated resource table (ART) entry for system administration and control of the resource while allocated. When implicit sharing is indicated by the absence of a specified usage mode, the appropriate linkage is established to coordinate concurrent access. The option is provided to allocate and open a resource with a single call to this function.

The base mode equivalent service is M_ASSIGN.

Entry Conditions

Calling Sequence

M.ASSN *rrsaddr* [,*cnpaddr*]

(or)

```
LA   R1,rrsaddr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'52'  (or)  M.CALL H.REXS,21
```

rrsaddr is the address of an RRS entry Type 1 through 6

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are time-out value, abnormal return address, options field, status field, and parameter link.

The option field contains an access and usage specification for opening this resource. This field is used only if the automatic open flag is set in the option word of the RRS. See the M.OPENR service.

If automatic open is indicated in the RRS, word 5 of the CNP must contain the address of a valid file control block (FCB) for this assignment. See the M.OPENR service.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R5	M.RTRN R5
(or)	(or)
M.RTRN R5 (CC1 set)	M.RTRN R5,R7 (CC1 set)

Registers

R5 contains the allocation index, a unique 32-bit integer number associated with the allocated resource. This index can set and release resource locks for exclusive or synchronous access.

R7 contains return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP (status values 25-29 are returned only when auto-open is indicated):

Value	Description
1	unable to locate resource (invalid pathname)
2	specified access mode not allowed
3	FAT/FPT space not available
4	blocking buffer space not available
7	static assignment to dynamic common
8	unrecoverable I/O error to volume
9	invalid usage specification
11	invalid RRS entry
12	LFC logically equated to unassigned LFC
13	assigned device not in system
14	resource already allocated by requesting task
15	SGO or SYC assignment by real-time task
17	duplicate LFC assignment attempted
19	invalid resource ID
20	specified volume not assigned
22	resource is marked for deletion
23	assigned device is marked off-line
24	segment definition allocation by unprivileged task
25	random access not allowed for this access mode
27	resource already opened in a different access mode
28	invalid access specification at open
29	specified LFC is not assigned to a resource for this task
38	time out occurred while waiting for resource to become available
46	unable to obtain resource descriptor lock (multiprocessor only)
50	resource is locked by another task
51	shareable resource is allocated in an incompatible access mode
54	unable to allocate resource for specified usage
55	ART space not available

Wait Conditions

When the resource is not available as indicated by status values 50-63, the task is placed in a wait state, as appropriate, if specified by a CNP.

6.2.5 M.ASYNCH - Set Asynchronous Task Interrupt

The M.ASYNCH service resets the asynchronous task interrupt mode back to the default environment.

The base mode equivalent service is M_ASYNCH

Entry Conditions

Calling Sequence

M.ASYNCH

(or)

SVC 1,X'1C' (or) M.CALL H.REXS,68

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set if asynchronous task interrupt already set

6.2.6 M.BACK - Backspace Record or File

The M.BACK service performs the following functions for backspacing records:

- if the file is actively generating output, the service issues a purge before the backspacing function. After the specified number of records are backspaced, the service returns control to the user.
- backspaces the specified number of records

M.BACK performs the following functions for blocked files:

- if the file is actively generating output, the service issues an end-of-file and purge before the backspace file function. Records are backspaced until an end-of-file record is found.
- backspaces the specified number of files
- the read/write control word then points to the end-of-file just encountered
- the M.BACK service cannot be used for SYC files or unblocked files

The base mode equivalent service is M_BACKSPACE.

Entry Conditions

Calling Sequence

M.BACK *fcbaddr* [R],[*number*]

(or)

```
LA    R1,fcbaddr
LNW   R4,number
SVC   1,X'35'  or  M.CALL H.IOCS,9   } (or)
SVC   1,X'36'  or  M.CALL H.IOCS,19  }
BIW   R4,$-1W
```

fcbaddr	is the FCB address
R	indicates to backspace by record (SVC1,X'35'). If omitted, backspace is by file (SVC1,X'36')
number	is the address of the word containing four times the number of records or files to backspace, or the contents of R4 if not supplied
$-1W	branches back to SVC until reaching the last word of R4

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO09	ILLEGAL OPERATION ON THE SYC FILE

6.2.7 M.BATCH - Batch Job Entry

The M.BATCH service submits a batch job stream located in a disk file. The disk file is described by the calling parameter in R1. Prior to calling this service, the specified disk file should be rewound to purge the contents of the blocking buffer if it has been dynamically built.

The base mode equivalent service is M_BATCH

Entry Conditions

Calling Sequence

M.BATCH *arga* [,*cnpaddr*]

(or)

```
LW   R1,arga
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'55' (or) M.CALL H.REXS,27
```

arga	is a PN vector, a PNB vector, or an RID vector for a permanent file; or an LFC or an FCB address for a temporary file
cnpaddr	is a CNP address or zero if CNP is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN R7 (CC1 set)
(or)	
M.RTRN R7 (CC1 set)	

Registers

R7	return status if a CNP is not supplied; otherwise, denial address

Status

CC1 set

Posted in R7 or in the status field of the CNP:

Value	Description
0	operation successful
1	pathname invalid
2	pathname consists of volume only
3	volume not mounted
4	directory does not exist
5	disk file has not been previously opened
6	unable to activate J.SSIN2, batch job not submitted
7	resource does not exist
14	unrecoverable I/O error while reading resource descriptor
18	unrecoverable I/O error while reading directory

6.2.8 M.BBTIM - Acquire Current Date/Time in Byte Binary Format

The M.BBTIM service acquires the system date and time in byte binary format. The date and time are returned in a two word buffer, the address of which is contained in the call. Refer to Appendix H for date and time formats.

This service can be executed in the IPU.

The base mode equivalent service is M_BBTIM.

Entry Conditions

Calling Sequence

M.BBTIM *addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'02000000'
SVC     2,X'50'  (or)  M.CALL H.REXS,74
```

addr is the address of a 2-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.9 M.BORT - Abort Specified Task, Abort Self, or Abort with Extended Message

6.2.9.1 M.BORT - Specified Task

This service allows the caller to abort another task. If the named task has been swapped out, it is not aborted until it regains CPU control. If the specified task is not in execution, the request is ignored.

The base mode equivalent service is M_BORT.

Entry Conditions

Calling Sequence

M.BORT *abcode,taskname*

(or)

```
LW   R5,abcode
ZR   R6                 } (or)  LD R6,taskname
LW   R7,taskno          }
SVC  1,X'56'  (or)  M.CALL H.REXS,19
```

abcode a 4 ASCII character abort code

taskname is the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the current task.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 Contains the task number, or zero if any of the following conditions exist:

- specified task was not found
- specified task name was multicopy
- owner name of the task requesting an abort is not privileged and is restricted from access to tasks with a different owner name
- task is exiting the system

6.2.9.2 M.BORT - Self

This service aborts the calling task by issuing an abort message, optionally performing a post-abort dump, and performing the functions common to the normal termination service.

Entry Conditions

Calling Sequence

M.BORT *abcode*

(or)

```
LW   R5,abcode
SVC  1,X'57'  (or)  M.CALL  H.REXS,20
```

abcode contains the 4-character ASCII abort code

Exit Conditions

Return Sequence

M.RTRN

Output Messages

task number `ABORT AT:` *xxxxxxxx - yyyyy mm/dd/yy hh:mn:ss zzzz*

task	is the 1- to 8-character name of the task being aborted
number	is the task number of the task being aborted
xxxxxxxx	is the location where the abort occurred
yyyyy	is the beginning of the DSECT
mm	is the month (2-character decimal number from 01 thru 12)
dd	is the day (2-character decimal number from 01 thru 31)
yy	is the year (2-character decimal number from 00 thru 99)
hh	is the hour (2-character decimal number from 00 thru 23)
mn	is the minutes (2-character decimal number from 00 thru 59)
ss	is the seconds (2-character decimal number from 00 thru 59)
zzzz	is the 4-character abort code

6.2.9.3 M.BORT - With Extended Message

A call to this service results in an abort of the specified task with an extended abort code message.

Entry Conditions

Calling Sequence

M.BORT *abcode,task,extcode*

(or)

```
LD   R2,extcode
LW   R5,abcode
LI   R6,0          } (or) LD R6,taskname
LW   R7,taskno     }
SVC  1,X'62'  (or) M.CALL  H.REXS,28
```

abcode	contains the abort code consisting of 4 ASCII characters
task	the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.
extcode	contains the extended abort code message consisting of 1 to 8 ASCII characters, left-justified and blank-filled

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if any of the following conditions exist:

- specified task was not found
- specified task name is multicopied
- owner name of the task requesting the abort is not privileged and is restricted from access to tasks with a different owner name (by the M.KEY file)
- task is exiting the system

Otherwise, contains the task number.

6.2.10 M.BRK - Break/Task Interrupt Link/Unlink

The M.BRK service allows the caller to clear the user break receiver or to establish the address of a routine to be entered when another task or the operator activates the caller task interrupt with an M.INT service.

The base mode equivalent service is M_BRK.

Entry Conditions

Calling Sequence

M.BRK *breakaddr*

(or)

```
LA   R7,breakaddr
SVC  1,X'6E'  (or)  M.CALL  H.REXS,46
```

breakaddr is the logical word address of the entry point of the task's break/task interrupt routine or zero to clear the break receiver

Exit Conditions

Return Sequence

M.RTRN

6.2.11 M.BRKXIT - Exit from Task Interrupt Level

This service must be called after executing the task interrupt routine. The M.BRKXIT transfers control back to the point of interruption by a task interrupt routine.

The base mode equivalent service is M_BRKXIT.

Entry Conditions

Calling Sequence

M.BRKXIT

(or)

SVC 1,X'70' (or) M.CALL H.REXS,48

Exit Conditions

Return Sequence

M.RTRN

6.2.12 M.BTIM - Acquire Current Date/Time in Binary Format

The M.BTIM service acquires the system date and time in binary format. The date and time are returned in a two word buffer, the address of which is specified in the call. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_BTIM.

Entry Conditions

Calling Sequence

M.BTIM *addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'01000000'
SVC     2,X'50'  (or)  M.CALL H.REXS,74
```

addr is the address of a 2-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.13 M.CLOSER - Close Resource

The M.CLOSER service terminates operations in the current access mode on a resource. The resource is marked closed in the file pointer table (FPT). The user-count in the appropriate allocated resource table (ART) entry is decremented if implicit shared use is in effect. For access modes other than read, the resource descriptor is updated. If last accessed functionality is enabled, the resource descriptor is also modified for read access mode. When the closing of a file implies a change of use or access mode for that resource, any tasks waiting for access to the resource in a compatible access mode are dequeued. If any logically equivalent resources are open, no further action is taken. For blocked files, any active output blocking buffer is purged. A close request to a resource that is already closed will result in an immediate return with the appropriate status posted.

The base mode equivalent service is M_CLOSER.

Entry Conditions

Calling Sequence

M.CLOSER *fcbaddr*[,*cnpaddr*]

(or)

```
LA   R1,fcbaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'43'  (or)  M.CALL H.REMM,22
```

fcbaddr is the address of a file control block (FCB)

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains return status if a CNP is not supplied; otherwise unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP

Value	Description
8	unrecoverable I/O error to volume
29	logical file code associated with FCB does not exist
31	resource was not open
46	unable to obtain resource descriptor lock of a multiprocessor

6.2.14 M.CLSE - Close File

The M.CLSE service marks a file closed in the file pointer table (FPT), and the count of open files (DFT.OPCT) is decremented. If any logically equivalent files are open, no further action is taken (for example, if count after decrementing is not equal to zero).

If the file is a system file or blocked file, any active output blocking buffers are purged. The file is marked closed by resetting the open bit in the file assignment table (FAT).

For files assigned to SYC or SGO, the current disk address updates the job table for job control.

This service issues an EOF prior to purging system files SLO and SBO which were opened for read/write. It also issues an EOF prior to purging for blocked files that are actively generating output.

The service ignores close requests to a file that is already closed.

The base mode equivalent service is M_CLSE.

Entry Conditions

Calling Sequence

M.CLSE *fcbaddr* [,[EOF] [,REW]]

(or)

```
LA     R1,fcbaddr
[SVC   1,X'38'  (or) M.CALL H.IOCS, 5]
[SVC   1,X'37'  (or) M.CALL H.IOCS, 2]
SVC    1,X'39'  (or) M.CALL H.IOCS,23
```

fcbaddr	is the FCB address
EOF	writes EOF (SVC 1,X'38'). See the M.WEOF description.
REW	rewinds file or device (SVC 1,X'37'). See the M.RWND description.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09    ILLEGAL OPERATION ON THE SYC FILE
IO38    WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE.
        A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF
        TASK HAS ONLY READ ACCESS TO UNIT.
```

6.2.15 M.CMD - Get Command Line

The M.CMD service returns the portion of the command line between the program name and the end of the line if the program name is specified on the command line. If data does not exist or the command line has already been issued, a null string is returned.

This service can be executed by the IPU.

The base mode equivalent service is M_CMD.

Entry Conditions

Calling Sequence

M.CMD

(or)

SVC 2,X'61' (or) M.CALL H.REXS,88

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6	contains the length of the string in bytes, if found; otherwise, zero
R7	contains the first byte address of the string, if found; otherwise, zero

6.2.16 M.CONABB - Convert ASCII Date/Time to Byte Binary Format

The M.CONABB service converts the system date and time from ASCII format to byte binary format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONABB.

Entry Conditions

Calling Sequence

M.CONABB *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'06000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 4-word buffer containing the ASCII-formatted date and time

outbuffer is the address of a 2-word buffer where the byte binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.17 M.CONADB - Convert ASCII Decimal to Binary

The M.CONADB service converts ASCII decimal doublewords to their binary equivalent.

An all blank doubleword converts to zero.

This service can be executed by the IPU.

The base mode equivalent service is M_CONADB.

Entry Conditions

Calling Sequence

M.CONADB [*addr*]

(or)

```
LD   R6,addr
SVC  1,X'28' (or) M.CALL H.TSM,7
```

addr is the address of a decimal number (left-justified, doubleword-bounded, ASCII-coded, blank-filled). If omitted, contents of R6 and R7 are converted.

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	contains zero if a character is nonnumeric
R7	contains the binary equivalent of the input

6.2.18 M.CONAHB - Convert ASCII Hexadecimal to Binary

The M.CONAHB service converts ASCII hexadecimal doublewords to their binary equivalent.

An all blank doubleword converts to zero.

This service can be executed by the IPU.

The base mode equivalent service is M_CONAHB.

Entry Conditions

Calling Sequence

M.CONAHB [*addr*]

(or)

```
LD   R6,addr
SVC  1,X'29'  (or)  M.CALL H.TSM,8
```

addr is the address of a hexadecimal number (left-justified, doubleword-bounded, ASCII-coded , blank-filled). If omitted, contents of R6 and R7 are converted.

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	contains zero if a character is not hexadecimal
R7	contains the binary equivalent of the input

6.2.19 M.CONASB - Convert ASCII Date/Time to Standard Binary

The M.CONASB service converts the system date and time from ASCII format to binary format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONASB.

Entry Conditions

Calling Sequence

M.CONASB *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'05000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 4-word buffer containing the ASCII formatted date and time

outbuffer is the address of a 2-word buffer where the binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.20 M.CONBAD - Convert Binary to ASCII Decimal

The M.CONBAD service converts binary words to their ASCII decimal equivalent.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBAD.

Entry Conditions

Calling Sequence

M.CONBAD [*addr*]

(or)

```
LW   R5,addr
SVC  1,X'2A'  (or)  M.CALL H.TSM,9
```

addr the address of a positive binary number

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6,R7 ASCII result, right-justified with leading ASCII zeros

6.2.21 M.CONBAF - Convert Binary Date/Time to ASCII Format

The M.CONBAF service converts the system date and time from binary format to ASCII format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBAF.

Entry Conditions

Calling Sequence

M.CONBAF *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'02000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 2-word buffer containing the binary formatted date and time

outbuffer is the address of a 4-word buffer where the ASCII formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.22 M.CONBAH - Convert Binary to ASCII Hexadecimal

The M.CONBAH service converts binary words to their ASCII hexadecimal equivalent.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBAH.

Entry Conditions

Calling Sequence

M.CONBAH [*addr*]

(or)

```
LW   R5,addr
SVC  1,X'2B'  (or)  M.CALL H.TSM,10
```

addr is the address of a binary number

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6,R7 ASCII result, right-justified with leading ASCII zeros

6.2.23 M.CONBBA - Convert Byte Binary Date/Time to ASCII

The M.CONBBA service converts the system date and time from byte binary format to ASCII format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBBA.

Entry Conditions

Calling Sequence

M.CONBBA *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'04000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 2-word buffer containing the byte binary formatted date and time

outbuffer is the address of a 4-word buffer where the ASCII formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.24 M.CONBBY - Convert Binary Date/Time to Byte Binary

The M.CONBBY service converts the system date and time from binary format to byte binary format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBBY.

Entry Conditions

Calling Sequence

M.CONBBY *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'01000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 2-word buffer containing the binary formatted date and time

outbuffer is the address of a 2-word buffer where the byte binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.25 M.CONBYB - Convert Byte Binary Date/Time to Binary

The M.CONBYB service converts the system date and time from the byte binary format to binary format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_CONBYB.

Entry Conditions

Calling Sequence

M.CONBYB *inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,=X'03000000'
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL H.REXS,75
```

inbuffer is the address of a 2-word buffer containing the byte binary formatted date and time

outbuffer is the address of a 2-word buffer where the binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.26 M.CONN - Connect Task to Interrupt

The M.CONN service indirectly connects a task to an interrupt level so that when the interrupt occurs, the specified task will be scheduled for execution or for the resumption of execution. If the specified task is not active, M.CONN will preactivate it. If preactivation is required, but the actual interrupt connection is denied, M.CONN deletes the residual task because the task would continue in the suspended state indefinitely.

The base mode equivalent service is M_CONN.

Entry Conditions

Calling Sequence

M.CONN *task,intlevel*

(or)

```
LW   R5,intlevel
LI   R6,0          } (or) LD R6,task (or)      LW  R6, PNV   }
LW   R7,taskno     }                           LI  R7, 0     }
SVC  1,X'4B' (or)  M.CALL H.REXS,10
```

task is the address of a doubleword containing the name of a task (left-justifed blank-filled 1- to 8-character ASCII) (system file only); or zero in word 0 and the task number in word 1; or pathname or RID vector in word 0 and zero in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

intlevel is the hardware priority level where the task is to be connected

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 denial code:

Value	Description
1	task already connected to an interrupt
2	another task connected to the specified interrupt
3	interrupt not SYSGEN specified indirectly connectable
4	specified task not found in dispatch queue or the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name

R7 zero if task not connected to interrupt; otherwise, contains the task number

6.2.27 M.CPERM - Create Permanent File

The M.CPERM service creates a permanent file. Permanent files are given names in directories and remain known to the operating system until explicitly deleted.

This service allocates a resource descriptor and the initial file space requirements for the file. Next, the specified attributes of the file are recorded in the resource descriptor. Finally, the name of the file is established in the indicated directory.

When a directory entry is established, it is linked to the resource descriptor of the file. This links the name of the file to the other attributes of the file. Typical file attributes are:

- file name
- file resource identifier (RID)
- file protection attributes
- file management attributes
- file initial space requirements

To create with possible multiple segments, the CNP address must be supplied. Byte 0 of the CNP option field contains the maximum number segments, allowed at creation. If M.CPERM creates a file with one segment which is less than the size requested, condition code bit 1 is set and status is returned in the CNP.

Asynchronous aborts and deletes are inhibited during execution of this service.

The base mode equivalent service is M_CREATEP.

Entry Conditions

Calling Sequence

M.CPERM *addr* [,*cnpaddr*] [,*rcbaddr*]

(or)

```
LW   R1,addr
LA   R2,rcbaddr (or) ZR  R2
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'20' (or) M.CALL H.VOMM,1
```

addr	contains a PN vector or PNB vector
cnpaddr	is the address of a CNP or zero if CNP not supplied
rcbaddr	is the address of an RCB or zero if default attributes are desired

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 Return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.28 M.CTIM - Convert System Date/Time Format

The M.CTIM service converts the system date and time from one of three standard formats to either of the other two formats. Refer to Appendix H for date and time formats. This service is callable from specific case macros that provide the function code in the macro call itself.

This service can be executed by the IPU.

The base mode equivalent service is M_CTIM.

Entry Conditions

Calling Sequence

M.CTIM *funct,inbuffer,outbuffer*

(or)

```
LA      R1,inbuffer
ORMW    R1,funct
LA      R2,outbuffer
SVC     2,X'51'  (or)  M.CALL  H.REXS,75
```

funct is the address of a word that contains the function code (see chart below) in byte 0, the most significant byte, and zeros in bytes 1, 2, and 3

inbuffer is the address of a 2- or 4-word buffer where the user provides the date and time, in any of the three standard formats, for the system to convert

outbuffer is the address of a 2- or 4-word buffer where the system returns the converted date and time values in the format requested

Funct code	Input format	Return format	Buffer Length in	Buffer Length out
1	Binary	Byte binary	2W	2W
2	Binary	Quad ASCII	2W	4W
3	Byte binary	Binary	2W	2W
4	Byte binary	Quad ASCII	2W	4W
5	Quad ASCII	Binary	4W	2W
6	Quad ASCII	Byte binary	4W	2W

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.29 M.CWAT - System Console Wait

The M.CWAT service suspends operation of the calling program until the specified I/O transfer is complete.

The base mode equivalent service is M_CWAT.

Entry Conditions

Calling Sequence

M.CWAT *tcpb*

(or)

```
LA   R1,tcpb
SVC  1,X'3D'  (or)  M.CALL H.IOCS,26
```

tcpb is the address of a type control parameter block (TCPB). See the Type Control Parameter Block section of chapter 5.

Exit Conditions

Return Sequence

M.RTRN

6.2.30 M.DASN - Deassign and Deallocate Resource

The M.DASN service deallocates a resource and disassociates it from a logical file code. When a device associated with an unformatted medium is detached, a message is issued to inform the operator to dismount the medium, unless the message was inhibited by user request or system constraints. Deallocation of a nonshared resource makes it available to other tasks. Deallocation of a shared resource makes the resource available if the caller is the last task to deallocate it or the access mode changes as a result of the deallocation to allow other compatible tasks to attach to the resource. Deallocation of SLO and SBO files results in their definitions being passed to the system output task for processing. If the specified logical file code was equated to other logical file codes in the system, only the specified LFC is deallocated. If a close directive was not issued, the resource is also closed.

The M.DASN service can also issue a dismount message for an unformatted medium with no resource deallocation.

The base mode equivalent service is M_DEASSIGN.

Entry Conditions

Calling Sequence

M.DASN *arga*[,*cnpaddr*]

(or)

```
LW   R1,arga
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'53'  (or)  M.CALL  H.REXS,22
```

arga is the address of the allocation index obtained when the resource was assigned

(or)

the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address, options field, and status field.

Setting bit 0 of the options field of the CNP specifies issue dismount message with no resource deallocation.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or in the status field of the CNP

Value	Description
8	unrecoverable I/O error to volume
29	logical file code associated with FCB not assigned
30	invalid allocation index
46	unable to obtain resource descriptor lock (multiprocessor only)

6.2.31 M.DATE - Date and Time Inquiry

The M.DATE service returns to the caller the date in ASCII, century, year, month and day, calendar information, and a count of the number of real-time clock interrupts since midnight. To aid in converting the interrupt count to time-of-day, counts of the number of interrupts per second and the number of interrupts per time unit are also returned.

This service can be executed by the IPU.

The base mode equivalent service is M_DATE.

Entry Conditions

Calling Sequence

M.DATE *pbaddr*

(or)

```
LA   R7,pbaddr
SVC  1,X'15'  (or)  M.CALL H.REXS,70
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

- Words 0 and 1 contain the current date in the format entered at IPL time.
- Word 2

Bytes	Contents in Binary
0	Century
1	Year
2	Month
3	Day

- Word 3 contains the number of clock interrupts since midnight
- Word 4 contains the number of clock interrupts per second; initialized by SYSGEN
- Word 5 contains the number of clock interrupts per time unit; initialized by SYSGEN

Exit Conditions

Return Sequence

M.IPURTN

6.2.32 M.DEBUG - Load and Execute Interactive Debugger

The M.DEBUG service causes one of the following events to occur:

- If the interactive debugger is currently loaded at the time the service is called, control is transferred to the debugger.
- If the interactive debugger is not currently loaded at the time the service is called, the debugger is loaded as an overlay segment, then control is transferred to the debugger.

If the task is nonbase and has a shared CSECT, the debugger is not loaded and error code 4 is returned. To debug a nonbase shared CSECT task, request the debugger at task activation.

The base mode equivalent service is M_DEBUG.

Entry Conditions

Calling Sequence

M.DEBUG

(or)

SVC 1,X'63' (or) M.CALL H.REXS,29

Exit Conditions

Return Sequence

M.RTRN R7

Registers

Normal Return to Debugger:

R7 contains the transfer address of the debugger if the debugger was loaded by this service call, or zero if the debugger was already loaded when this service was called

Abnormal Return Sequence to Caller:

R7

Value	Description
2	debugger load module not found
4	invalid preamble
5	insufficient task space for loading
6	I/O error on resource descriptor
7	I/O error on resource
8	loading error

6.2.33 M.DEFT - Change Defaults

The M.DEFT service changes the caller's working current directory or project group protection or both.

This service should be called with two separate calls if both project group protection and the current working directory are being changed.

When both project group and working directory are specified in a single call, the project group is changed before the current working directory is changed. After changing the project group, if the attempt to establish the current working directory fails, the new project group protection will remain in effect and the caller will be notified through an error status code that the current working directory request failed. The caller must determine whether to continue with the new project group or to reestablish another project group.

The base mode equivalent service is M_DEFT.

Entry Conditions

Calling Sequence

M.DEFT [*vector*] [,*cnpaddr*] [,*prjaddr*] [,*keyaddr*]

(or)

```
LW   R1,vector
LA   R4,prjaddr (or) ZR  R4
LA   R5,keyaddr (or) ZR  R5
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'27' (or) M.CALL H.VOMM,8
```

vector	contains a PN vector, PNB vector, or RID vector
cnpaddr	is a CNP address or zero if a CNP is not supplied
prjaddr	is the address of the new project group name or zero if the project group name will not be changed
keyaddr	is the address of the new project group key or zero if a key is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or) M.RTNA (CC1 set)	(or) M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.34 M.DELR - Delete Resource

The M.DELR service explicitly deletes volume resources (i.e., directories, files, and memory partitions). A directory may be deleted only if it is empty. The caller must have delete access to the resource in order to delete it.

First, this service deletes the directory entry for the specified resource. Next, the volume space requirements are released. Finally, the resource descriptor is released.

If the resource is allocated at the time of the delete request, only the directory entry is deleted. The volume space requirements and the resource descriptor for the resource will be released when the last assignment to the resource is removed.

To delete a permanent file or memory partition, the pathname or pathname block must be supplied. To delete a directory, the pathname or pathname block must be supplied and all files which were defined in the directory must have been previously deleted.

To delete a temporary file, resource identifier (RID), the logical file code (LFC), or the address of a file control block (FCB) must be supplied.

Asynchronous abort and delete are inhibited during execution of this service.

The base mode equivalent service is M_DELETER.

Entry Conditions

Calling Sequence

M.DELR [*addr*] [,*cnpaddr*]

(or)

```
LW   R1,addr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'24'  (or)  M.CALL H.VOMM,5
```

addr	contains a PN vector, a PNB vector, an LFC, or an FCB
cnpaddr	is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.35 M.DELTSK - Delete Task

The M.DELTSK service forces I/O completion and immediately aborts the specified task. This service should be used only when abort fails to remove a task or when a task is queued for a resource. File integrity can be affected because operations are not allowed to complete normally. To preserve system integrity, the kill directive is processed as an abort for the amount of time specified by KTIMO (SYSGEN). If this does not remove the task, it is killed.

The base mode equivalent service is M_DELTSK.

Entry Conditions

Calling Sequence

M.DELTSK *abcode,task,extcode*

(or)

```
LD   R2,extcode
LW   R5,abcode
LI   R6,0          } (or) LD R6,taskname
LW   R7,taskno     }
SVC  1,X'5A'  (or) M.CALL  H.REXS,31
```

abcode contains the 4-character ASCII abort code

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

extcode contains the 1- to 8-character ASCII extended abort code message, left-justified and blank-filled.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains the task number, zero if the specified task was not found or if the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name

Output Messages

Modifies abort message to:

task number `ABORT AT:` *xxxxxxxx - yyyy mm/dd/yy hh:mn:ss zzzzzzzz*

task	is the 1- to 8-character name of the task being aborted
number	is the task number of the task being aborted
xxxxxxxx	is the location the abort occurred
yyyy	is the bias to the logical start of the task
mm	is the month (2-character decimal number from 01 thru 12)
dd	is the day (2-character decimal number from 01 thru 31)
yy	is the year (2-character decimal number from 00 thru 99)
hh	is the hour (2-character decimal number from 00 thru 23)
mn	is the minutes (2-character decimal number from 00 thru 59)
ss	is the seconds (2-character decimal number from 00 thru 59)
zzzzzzzz	is the extended message code supplied with the call to this service

6.2.36 M.DEVID - Get Device Mnemonic or Type Code

The M.DEVID service allows the user to pass a device mnemonic or a generic device type code and receive the corresponding type code or mnemonic. Device mnemonic and device type codes are listed in Appendix A.

This service can be executed by the IPU.

The base mode equivalent service is M_DEVID.

Entry Conditions

Calling Sequence

M.DEVID *id*

(or)

```
LW   R2,id
SVC  1,X'14' (or)  M.CALL   H.REXS,71
```

id is a word containing either a device mnemonic in the right halfword and in the left zero halfword or a device type code in byte 3 and zero in bytes 0-2

Exit Conditions

Return Sequence

M.IPURTN 2

Registers

Registers if input was a device mnemonic:

R2 bytes 0-2 contain zeros
byte 3 contains the corresponding device type code

Registers if input was a device type code:

R2 left halfword contains zero
right halfword contains the corresponding device mnemonic

Registers if input was a mnemonic or device type code not in the system device type table (DTT):

R2 bit 0 is set
bits 1-31 are unchanged

6.2.37 M.DIR - Create Directory

The M.DIR service creates a permanent directory. Permanent directories are given names in the root directory and remain known to MPX-32 until explicitly deleted.

Directories contain the names of permanent files and memory partitions that are created in the directories.

This service allocates a resource descriptor and the volume space requirements for the directory. Next, the service records the indicated attributes of the directory in the resource descriptor. Finally, the service establishes the name of the directory in the indicated previous level, or parent directory.

When the directory is established, the directory entry is linked to the resource descriptor of the new directory. This links the name of the new directory to the other attributes of the new directory. Typical directory attributes are:

- directory name
- directory resource identifier (RID)
- directory protection attributes
- directory management attributes
- directory volume space requirements

Asynchronous abort and delete are inhibited during execution of this service.

The base mode equivalent service is M_DIR.

Entry Conditions

Calling Sequence

M.DIR [[ARGA=]*vector*] [,[CNP=]*cnpaddr*] [,[RCB=]*rcbaddr*]

(or)

```
LW   R1,vector
LA   R2,rcbaddr (or) ZR R2
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'23' (or) M.CALL H.VOMM,4
```

vector	contains a PN vector or PNB vector
cnpaddr	is a CNP address or zero if CNP not supplied
rcbaddr	is an RCB address or zero if default attributes are desired

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.38 M.DISCON - Disconnect Task from Interrupt

The M.DISCON service disconnects a task from a centrally connected interrupt level.

The base mode equivalent service is M_DISCON.

Entry Conditions

Calling Sequence

M.DISCON *task*

(or)

```
LI    R6,0          }
LW    R7,taskno     } (or)  LD R6,taskname
SVC   1,X'5D'  (or)  M.CALL  H.REXS,38
```

task is the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Normal Return Sequence:

M.RTRN 7

Registers

R7 contains the task number

Abnormal Return Sequence:

M.RTRN 6,7

Registers

R6 contains denial code as follows:

Value	Description
0	no error
1	task not found in dispatch queue or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file.
2	task not indirectly connected
3	task connected to an invalid interrupt

R7 zero if the task was not previously found or was not connected to an interrupt level

6.2.39 M.DLTT - Delete Timer Entry

The M.DLTT service resets the timer for the specified task so that its specified function is no longer performed upon time out. Deletion of the timer entry does not delete the associated task. One-shot timers are deleted on expiration.

The base mode equivalent service is M_DLTT.

Entry Conditions

Calling Sequence

M.DLTT *timer*

(or)

```
LW   R7,timer
SVC  1,X'47'  (or)  M.CALL H.REXS,6
```

timer is the right-justified 2-character ASCII name of a timer

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set timer entry not found

6.2.40 M.DMOUNT - Dismount Volume

The M.DMOUNT service performs a logical dismount and, optionally, a physical dismount of a user volume.

For a nonpublic volume, M.DMOUNT decrements the use count for the volume in the mounted volume table (MVT) entry, provided that the requestor has no resources allocated on the volume. A physical dismount is performed if the MVT use count is zero. Otherwise, the physical dismount is pending, and all mount requests for the volume are denied.

For a physical dismount of a public volume, M.DMOUNT establishes a use count in the MVT based on the total number of resources allocated on the volume at the time the dismount request is made. When the use count is zero, a physical dismount is performed. M.DMOUNT confirms the completed dismount with the system operator through the system console, as specified in the CNP.

Only the system administrator can request the dismount of a public volume.

The base mode equivalent service is M_DISMOUNT.

Entry Conditions

Calling Sequence

M.DMOUNT *voladdr* [,*cnpaddr*]

(or)

```
LA   R1,voladdr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'4A'  (or)  M.CALL  H.REMM,19
```

voladdr specifies the address of a doubleword bounded field containing a volume name

cnpaddr specifies the address of a caller notification packet, if notification is desired.

Applicable portions of the CNP for this function are option field, abnormal return address, status field, and Word 3.

Option field: (Word 2 of CNP)

Bit	Meaning if set
0	physical dismount requested
1	no logical dismount
2	public volume dismount request
3	inhibit operator interaction

Note: Bit 3 is ignored if the volume was mounted with operator intervention inhibited.

Word 3: Dismount device specification, if the option bit is set

Byte	Definition
1	device type code
2	channel address
3	subchannel address

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or in the status field of the CNP:

Value	Description
14	caller has outstanding resource assignments on this volume
20	volume not assigned to this task or volume is public
86	cannot dismount the system volume
87	unable to dismount public volume because compatible mode public volume dismount (CMPMM) option was specified at SYSGEN
88	unable to dismount public volume. SA attribute required
89	unable to dismount public volume. Missing CNP option on dismount request

6.2.41 M.DSMI - Disable Message Task Interrupt

The M.DSMI service disables the task interrupts for messages sent to the calling task. M.DSMI is useful for synchronization gating of the task message interrupts.

This service can be executed by the IPU.

The base equivalent service is M_DSMI.

Entry Conditions

Calling Sequence

M.DSMI

(or)

SVC 1,X'2E' (or) M.CALL H.REXS,57

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1	task interrupts were already disabled

6.2.42 M.DSUB - Disable User Break Interrupt

The M.DSUB service deactivates the user break interrupt activated with the M.ENUB service and allows user breaks by the terminal break key to be acknowledged.

This service can be executed by the IPU.

The base equivalent service is M_DSUB.

Entry Conditions

Calling Sequence

M.DSUB

(or)
SVC 1,X'12' (or) M.CALL H.REXS,73

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1 set user break already disabled

6.2.43 M.DUMP - Memory Dump Request

The M.DUMP service dumps the caller's program status doubleword (PSD), general purpose registers, and specified memory limits to the SLO file. Start and end addresses are truncated to the nearest eight-word boundaries and memory is dumped between the truncated limits.

The format of the dump is side-by-side hexadecimal with ASCII. The PSD and registers precede the specified memory limits. The PSD and registers are extracted from the first level of push-down of the calling task. Optionally, R5 can specify the address of a ten word block containing R0 through R7 and the PSD to be dumped, respectively. Any task can request a memory dump.

The base mode equivalent service is M_DUMP.

Entry Conditions

Calling Sequence

M.DUMP *start,end* [,*mem3*]

(or)

```
ZR   R5 (or)  LA   R5,mem3
LW   R6,start
LW   R7,end
SVC  1,X'4F'  (or)  M.CALL H.REXS,12
```

start	is the low logical word address requested in dump
end	is the high logical word address requested in dump
mem3	is the optional address of ten consecutive words containing R0 through R7 and a PSD, respectively. If R5 is zero, the registers and PSD dumped are taken from the current stack frame locations.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 is unchanged, or contains an error message as follows:

Value	Description
1	high dump limit less than low limit
4	no FAT or FPT space available
5	request made with insufficient levels of push-down available
6	cannot allocate SLO file
7	unrecoverable I/O error

R7 is unchanged, or contains zero if dump could not be performed

6.2.44 M.EAWAIT - End Action Wait

The M.EAWAIT service waits for the completion of no-wait request or I/O end action if any are queued. If no such requests or actions are outstanding, the service returns immediately to the user. This service is similar to the M.ANYW service.

The base mode equivalent service is M_AWAITACTION.

Entry Conditions

Calling Sequence

M.EAWAIT *time*

(or)

```
LW   R6,time
SVC  1,X'1D'   (or)   M.CALL H.EXEC,40
```

time contains the negative number of time units to elapse before the wait is terminated, or zero if wait for an indefinite period is requested.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

MS31 USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN END-ACTION ROUTINE

6.2.45 M.ENMI - Enable Message Task Interrupt

The M.ENMI service enables task interrupts for messages sent to the calling task. It removes an inhibit condition previously established by invoking the M.DSMI service.

The base mode equivalent service is M_ENMI.

Entry Conditions

Calling Sequence

M.ENMI

(or)

SVC 1,X'2F' (or) M.CALL H.REXS,58

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set task interrupts were already enabled

6.2.46 M.ENUB - Enable User Break Interrupt

The M.ENUB service activates the user break interrupt and causes further user breaks from the user terminal break key to be ignored.

This service can be executed by the IPU.

The base mode equivalent service is M_ENUB.

Entry Conditions

Calling Sequence

M.ENUB

(or)

SVC 1,X'13' (or) M.CALL H.REXS,72

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1 set user break already enabled

6.2.47 M.ENVRMT - Get Task Environment

The M.ENVRMT service obtains more information on the task environment than what is provided in the task option word.

This service can be executed by the IPU.

The base mode equivalent service is M_ENVRMT.

Entry Conditions

Calling Sequence

M.ENVRMT

(or)

SVC 2,X'5E' (or) M.CALL H.REXS,85

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains the task environment word as follows:

Bit	Definition
0	0 if batch task or real-time task 1 if interactive
1	0 if option NOCOMMAND is set 1 if option COMMAND is set
2	0 if option NOERR is set 1 if option ERROR is set
3	0 if cataloged or linked unprivileged 1 if cataloged or linked privileged
4	0 if currently unprivileged 1 if currently privileged
5	0 if TSA is not moved 1 if TSA is moved
6	0 if MPX-32 is mapped into task address space 1 if MPX-32 is mapped out of task address space (CONCEPT 32/2000)
7	0 if task is not in demand page mode 1 if task is in demand page mode (CONCEPT 32/2000 only)
8	0 if task is not in segment register mode 1 if task is in segment register mode (CONCEPT 32/2000 only)
9-31	reserved

6.2.48 M.EXCLUDE - Exclude Memory Partition

The M.EXCLUDE service allows a nonbase task to dynamically deallocate any common area previously included. This service causes the assign count and user count to be decremented. The partition area is deleted and its resources returned to the free list when the assign count goes to zero. This service is also called by the exit processor (H.REMM,3) when a task aborts or ends abnormally while associated with a memory partition. The partition is identified by the allocation index obtained when the partition was included, or by the 1- to 8-character name used to create and with the 1- to 8-character owner name or task number used to include it.

The base mode equivalent service is M_EXCLUDE.

Entry Conditions

Calling Sequence

M.EXCLUDE [*arga*],[*cnpaddr*][,*argb*]

(or)

```
LW   R1,arga
LD   R4,argb
LA   R7,cnpaddr  (or)  ZR R7
SVC  2,X'41'  (or)  M.CALL  H.REMM,14
```

arga contains a PN vector, PNB vector, the address of the 1- to 8-character left-justified, blank-filled system partition name, or the allocation index obtained when the partition was included.

argb is the address of a doubleword containing a left-justified task number in word 0 and zero in word 1

(or)

a 1- to 8-character left-justified, blank-filled owner name used to include the partition

(or)

not required, if an allocation index is used

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
5	shared memory table (SMT) entry not found for this partition
30	invalid allocation index
35	attempt to exclude memory partition that is not mapped into requesting task's address space
39	unable to write back data section
58	shared memory table space is not available

6.2.49 M.EXIT - Terminate Task Execution

The M.EXIT service performs all normal termination functions required of exiting tasks. All devices and memory are deallocated, related table space is erased, and the task's dispatch queue entry is cleared.

The base mode equivalent service is M_EXIT.

Entry Conditions

Calling Sequence

M.EXIT

(or)

SVC 1,X'55' (or) M.CALL H.REXS,18

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

RX92 TASK HAS ATTEMPTED NORMAL EXIT WITH MESSAGES IN ITS RECEIVER QUEUE

6.2.50 M.EXTD - Extend File

The M.EXTD service allows the space of a file to be manually extended. The caller can specify the size of the requested extension or to use the default file extension parameters defined when the file was created. If the specified size cannot be obtained, M.EXTD tries to extend by the maximum extension size that was specified at resource creation. If that size cannot be obtained, M.EXTD tries to extend by the minimum extension size that was specified at resource creation. A CC1 indicates that the requested extension size is not obtained. If the file was created with the zero option specified, the extension is zeroed.

This service extends only temporary or permanent files that are manually extendable. Directories and memory partitions cannot be extended. The caller must have write, update, or append access to extend the file.

The caller can extend a file regardless of whether the file is currently allocated. Additionally, the caller can supply any allowable resource specification, for example, pathname (PN), pathname block (PNB), resource ID (RID), logical file code (LFC) or address of a file control block (FCB).

Asynchronous abort and delete are inhibited for execution of this entry point.

The base mode equivalent service is M_EXTENDFILE.

Entry Conditions

Calling Sequence

M.EXTD [*addr*],[*cnpaddr*][,*blocks*]

(or)

```
LW   R1,addr
LW   R6,blocks (or) ZR  R6
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'25' (or) M.CALL  H.VOMM,6
```

addr	contains a PN vector; a PNB vector, an RID vector, an LFC, or an FCB address
cnpaddr	is a CNP address or zero if CNP not supplied
blocks	is an address containing the number of blocks to extend the file by or zero if RCB extension parameters specified during file creation are to be used

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R6	M.RTRN R6
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R6	contains the number of contiguous blocks file actually extended by
R7	contains the return status codes if a CNP is not supplied. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.51 M.FD - Free Dynamic Extended Indexed Data Space

The M.FD service allows a task to deallocate the most recently acquired extended memory map block, thus contracting the task's address space.

Entry Conditions

Calling Sequence

M.FD

(or)

SVC 1,X'6A' (or) M.CALL H.REMM,9

Exit Conditions

Return Sequence

M.RTRN R3

Registers

R3 contains the new upper limit of extended memory, or zero if no extended memory is left allocated, M.MEMB service is in use, or a multicopied shared image is in use

6.2.52 M.FE - Free Dynamic Task Execution Space

The M.FE service allows a task to dynamically deallocate the most recently acquired execution space map block, thus contracting the task's address space.

Entry Conditions

Calling Sequence

M.FE

(or)

SVC 1,X'68' (or) M.CALL H.REMM,11

Exit Conditions

Return Sequence

M.RTRN R3 (or) abort user with RM76

Registers

R3 contains the new upper address of execution space

Abort Cases

RM76 USER ATTEMPTED DEALLOCATION OF TSA

6.2.53 M.FWRD - Advance Record or File

The M.FWRD service performs the following functions for advancing records:

- verifies volume record if BOT is encountered on multivolume magnetic tape
- advances specified number of records

M.FWRD performs the following functions for blocked files.

- advance logical records until an end-of-file is found. The read/write control word points to the first record after the end-of-file.
- verifies volume record if BOT is encountered on multivolume magnetic tape
- advances specified number of files

The M.FWRD service is not applicable for SYC files or unblocked files.

The base mode equivalent service is M_ADVANCE.

Entry Conditions

Calling Sequence

M.FWRD *fcbaddr*,[R],[*number*]

(or)

```
LA      R1,fcbaddr
LNW     R4,number
SVC       1,X'33'  (or)  M.CALL H.IOCS,7   } (or)
SVC       1,X'34  (or)  M.CALL H.IOCS,8    }
BIB     R4,$-1W
```

fcbaddr	is the FCB address
R	specified advance by record (SVC1,X'33'). If not specified, the default is advance by file (SVC1,X'34').
number	is the address of the word containing the number of records or files to be advanced, or the contents of R4 if not supplied
$-1W	branches back to SVC the number of times specified by *number*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

I006	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
I007	THE TASK HAS ATTEMPTED TO PERFORM AN OPERATION WHICH IS NOT VALID FOR THE DEVICE TO WHICH THE USER'S FILE IS ASSIGNED (E.G., A READ OPERATION SPECIFIED FOR A FILE ASSIGNED TO THE LINE PRINTER)
I009	ILLEGAL OPERATION ON THE SYC FILE
I030	ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE

Output Messages

Mount/dismount messages are displayed if EOT is encountered on multivolume magnetic tape.

6.2.54 M.GADRL - Get Address Limits

The M.GADRL service returns the logical addresses associated with the boundaries of a nonbase mode task. The address returned in R5 reflects any increments previously allocated using the M.GE service and the address returned in R7 reflects any increments previously allocated using the M.GD service. These addresses do not reflect any increments previously allocated using the M.INCL, M.INCLUDE, or M.SHARE services.

Entry Conditions

Calling Sequence

M.GADRL

(or)

SVC 1,X'65' (or) M.CALL H.REXS,41

Exit Conditions

Return Sequence

M.RTRN 3,4,5,6,7

Registers

R3	contains the logical word address of the first location of the task's DSECT (always on a page boundary)
R4	contains the logical word address of the last location in the DSECT actually loaded by the loader plus one word
R5	contains the logical word address of the last location currently available in the task's contiguously allocated DSECT (always a map block boundary minus one word)
R6	contains the logical word address of the first location of the task's CSECT or COMMON allocation (always a map block boundary)
R7	contains the logical word address of the last location currently available in the task's contiguously allocated extended indexed data space (always a map block boundary minus one word). If the M.GD service has not already been called, R5 contains the word address of the last word of non-extended address space.

6.2.55 M.GADRL2 - Get Address Limits

The M.GADRL2 service returns the logical addresses associated with the boundaries of a nonbase mode task. R3 contains any increments previously allocated using the M.GE service. R5 contains any increments previously allocated by the M.GD service. R6 contains increments previously allocated by the M.INCL, M.INCLUDE, or M.SHARE services. R7 contains the upper address boundary of the task.

R6 and R7 return addresses not available from M.GADRL and give boundaries for extended memory that reflect the use of the CATALOG or TSM SPACE directive and the presence of extended MPX-32 or other partitions mapped into the logical extended task address space.

Entry Conditions

Calling Sequence

M.GADRL2

(or)

SVC 2,X'7B' (or) M.CALL H.REXS,80

Exit Conditions

Return Sequence

M.RTRN 1,2,3,4,5,6,7

Registers

R1	contains the logical word address of the first location of the task's DSECT (always on a page boundary)
R2	contains the logical word address of the last location in the DSECT actually loaded by the loader, plus one word
R3	contains the logical word address of the last location currently available in the task's contiguously allocated DSECT (always a map block boundary minus one word)
R4	contains the logical word address of the first location of the task's CSECT or COMMON allocation (always a map block boundary)
R5	contains the logical word address of the last location currently available in the task's contiguously allocated extended indexed data space (always a map block boundary minus one word). If the M.GD service has not already been called, then R5 will contain the word address of the last word of non-extended address space.
R6	contains the logical word address of the first allocated location of extended address space which was obtained by inclusion via the M.INCL, M.INCLUDE, or M.SHARE services (always a map block boundary). If no extended maps have been included, R6 will equal R7.
R7	contains the logical word address of the last location in the task's logical address space (always a map block boundary minus one word)

6.2.56 M.GD - Get Dynamic Extended Data Space

The M.GD service allows the nonbase mode task to dynamically acquire an additional map block of memory in its extended area. The memory is of the same type specified when the task was cataloged. It is mapped in a logically contiguous manner, with the first request map starting at 128KW. The task can call this service up to 192 times, if sufficient memory exists, to expand its extended indexed data space. Alternately, the task can choose to deallocate this space in the reverse order by M.FD. The task is suspended until the allocation is successful.

Memory is allocated in 2KW increments.

Entry Conditions

Calling Sequence

M.GD

(or)

SVC 1,X'69' (or) M.CALL H.REMM,8

Exit Conditions

Return Sequence

M.RTRN R3,R4

Registers

R3	contains the logical address of the allocated memory, or zero if allocation conflicts occurred
R4	contains ending logical word address of allocated memory or error code if R3 is zero

Error Condition

R3 equals zero

R4

Value	Description
1	attempted allocation of an excessive number of map blocks
2	attempted allocation exceeds physical memory configured
3	M.MEMB service in use

6.2.57 M.GDD - Get Dynamic Extended Discontiguous Data Space

The M.GDD service allows a nonbase mode task to acquire an additional map block of memory in its extended address space. The memory is mapped in logically ascending order starting at the extended address origin. When the next contiguous map block is shared, the map acquired is logically discontiguous with previous M.GDD allocations. The M.FD service is used to deallocate the memory.

Exclusion of shared memory which is logically below the last map allocated by this service creates an empty gap in the logical address space which does not get allocated by a subsequent call to this service. The task can call this service repeatedly until it has expanded its space to the logical limit (i.e., limit varies with use of SPACE directive). The task is queued until the allocation is successful.

This service reports an error if the M.MEMB service is in use or if the task has included a multicopied shared image. If a multicopied shared image is included a denial will be returned from the M.FD service also and exclusion of the image does not enable the use of either M.GDD or M.FD.

Entry Conditions

Calling Sequence

M.GDD

(or)

SVC 2,X'7C' (or) M.CALL H.MEMM,9

Exit Conditions

Return Sequence

M.RTRN 3,4

Registers

R3	logical address of allocated memory
R4	ending logical word address of allocated memory or error code if R3 is zero

Error Condition

R3 equals zero

R4

Value	Description
1	attempted allocation of an excessive number of map blocks
2	attempted allocation exceeds physical memory configured
3	M.MEMB service in use
4	Multicopied shared image is included

6.2.58 M.GE - Get Dynamic Task Execution Space

The M.GE service allows the nonbase mode task to dynamically expand its memory allocation in map block increments, starting at the end of its DSECT up to the end of its logical address space. The additional memory is of the same type specified when the task was cataloged. The task is mapped in a logically contiguous manner up to the start of its CSECT or Global common, or 128KW, whichever occurs first. The task is suspended until the allocation is successful.

Memory is allocated in 2KW increments.

Entry Conditions

Calling Sequence

M.GE

(or)

SVC 1,X'67' (or) M.CALL H.REMM,10

Exit Conditions

Return Sequence

M.RTRN R3,R4

Registers

R3 contains the starting logical address of the new map block

R4 contains the ending logical address of the new map block

Error Condition

R3 equals zero

R4

Value	Description
1	attempted allocation of an excessive number of map blocks
2	attempted allocation exceeds physical memory configured
3	M.MEMB service in use

6.2.59 M.GETDEF - Get Definition for Terminal Function

The M.GETDEF service returns, for the requested terminal function, an appropriate string of bytes in the specified buffer and indicates the length of the returned string. The user must specify the LFC of a terminal that is currently open and allocated, the buffer address and a terminal function. For this service to operate, the partition TERMPART must exist and have been initialized by J.TDEFI. For more information, refer to MPX-32 Reference Manual, Volume II, Chapter 11.

The base mode equivalent service is M_GETDEF.

Entry Conditions

Calling Sequence

M.GETDEF [*tdefiblk*]

(or)

```
LA   R1,tdefiblk
SVC  2,X'7A' (or)  M.CALL H.TSM,15
```

tdefiblk is the logical 24-bit word address of the first word of the TERMDEF information block formatted as follows:

Word	Description
0	open and allocated terminal's LFC
1	user buffer 24 bit address for returned information
2	half word 0 is the requested function (2 ASCII alphanumeric characters); half word 1 is reserved.
3	half word 0 is the user buffer length in bytes; half word 1 is the length in bytes of string returned by the service to the user's buffer
4	optional X coordinate for cursor positioning functions
5	optional Y coordinate for cursor positioning functions

R1 is assumed to contain the address if *tdefiblk* is not supplied.

Exit Conditions

Return Sequence

M.RTRN — On normal completion the string for the requested function is in the user's buffer, and the length of this string is in the TERMDEF information block string length field.

CC1 set — Error detected. The string length in the TERMDEF information block is set to 0 and the function contains the error number with the following meanings:

function=error

error	Description
1	invalid LFC supplied
2	unknown terminal type
3	user buffer is too large (>2K)
4	cannot include partition
5	undefined function requested
6	user buffer is too small
7	partition data integrity suspect
8	invalid terminal type supplied
9	invalid user buffer address
10	function is invalid for this terminal
11	TERMDEF is not installed
N/A	TERMDEF information block address is invalid (CC1 set only)

Registers

R1 — unchanged; CC1 set if an error is detected.

Abort Cases

MF01 — A MAP FAULT TRAP HAS OCCURRED. THIS IS THE RESULT OF A BAD MEMORY REFERENCE OUTSIDE OF THE USER'S ADDRESSABLE SPACE.

6.2.60 M.GMSGP - Get Message Parameters

The M.GMSGP service is called from the message receiver routine of a task that has received a message interrupt. It transfers the message parameters into the designated receiver buffer, and posts the owner name and task number of the sending task into the parameter receive block (PRB).

The base mode equivalent service is M_GMSGP.

Entry Conditions

Calling Sequence

M.GMSGP *prbaddr*

(or)

```
LA   R2,prbaddr
SVC  1,X'7A'  (or)  M.CALL  H.REXS,35
```

prbaddr is the logical address of the parameter receive block (PRB)

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing status error code:

Value	Description
0	normal status
1	invalid PRB address
2	invalid receiver buffer address or size detected during parameter validation
3	no active send request
4	receiver buffer length exceeded during transfer

6.2.61 M.GRUNP - Get Run Parameters

The M.GRUNP service is called by a task that received a run request. It transfers the run parameters into the designated receiver buffer, and posts the owner name and task number of the sending task into the Parameter Receive Block (PRB).

The base mode equivalent service is M_GRUNP.

Entry Conditions

Calling Sequence

M.GRUNP *prbaddr*

(or)

```
LA   R2,prbaddr
SVC  1,X'7B'  (or)  M.CALL H.REXS,36
```

prbaddr is the logical address of the Parameter Receive Block (PRB)

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing status error code:

Value	Description
0	normal status
1	invalid PRB address
2	invalid receiver buffer address or size detected during parameter validation
3	no active send request
4	receiver buffer length exceeded during transfer

6.2.62 M.GTIM - Acquire System Date/Time in Any Format

The M.GTIM service acquires the system date and time in binary, byte binary, or quad-ASCII format, as described in Appendix H. The system date/time are also retrievable by using any of the three specific case macros. These macros generate the same SVC call, but the function code is provided by the macro.

This service can be executed by the IPU.

The base mode equivalent service is M_GTIM.

Entry Conditions

Calling Sequence

M.GTIM *funct,addr*

(or)

```
LA      R1,addr
ORMW    R1,funct
SVC     2,X'50'  (or)  M.CALL  H.REXS,74
```

funct is the address of a word containing the function code (see chart below) in byte 0, most significant byte, and zeros in bytes 1, 2, and 3

addr is the address of a buffer where the service places the date and time in the format requested. This buffer is two or four words in length depending on the format.

Function Code	Return Format	Buffer Length
1	binary	2W
2	byte binary	2W
3	quad ASCII	4W

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 byte 0 contains the function code and bytes 1-3 contain the buffer address as used by the call. All other are returned intact.

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

6.2.63 M.GTSAD - Get TSA Start Address

The M.GTSAD service returns the TSA starting address of the calling task.

This service is callable by the IPU.

The base mode equivalent service is M_GTSAD.

Entry Conditions

Calling Sequence

M.GTSAD

(or)

SVC 2,X'7D' (or) M.CALL H.REXS,91

Exit Conditions

Return Sequence

M.IPURTN R1

Registers

R1	logical address of the TSA of the caller

6.2.64 M.HOLD - Program Hold Request

The M.HOLD service makes the specified task ineligible for CPU control by setting the hold bit in the CPU dispatch queue. The specified task remains on hold until the user issues the OPCOM CONTINUE directive. If the specified task is not in the CPU dispatch queue, the request is ignored.

The base mode equivalent service is M_HOLD.

Entry Conditions

Calling Sequence

M.HOLD *task*

(or)

```
LI    R6,0          } (or)  LD R6,taskname
LW    R7,taskno     }
SVC   1,X'58'  (or)  M.CALL  H.REXS,25
```

task is the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the task number, or zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name with the M.KEY file.

6.2.65 M.ID - Get Task Number

The M.ID service allows the user to pass the address of a parameter block containing any of the following: task number, task load module name, owner name, task pseudonym. The service provides the missing items if a matching entry is found. Initially, the caller passes zero (0) as the index value following the parameter block address. If more than one task in the dispatch queue satisfies the given parameters, the service returns to the caller with an index value in R5 for retrieval of further entries. The caller is responsible for updating the index with the contents of R5 and reissuing M.ID until all tasks that meet specifications have been identified or R5 equals zero.

The base mode equivalent service is M_ID.

Entry Conditions

Calling Sequence

M.ID *pbaddr,index*

(or)

```
LW   R5,index
LA   R7,pbaddr
SVC  1,X'64'  (or)  M.CALL  H.REXS,32
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

Word	Contents
0	task activation sequence number
1-2	task load module name
3-4	owner name
5-6	pseudonym

The user supplies those items that are known and zeros the other words.

index is a variable equal to zero for initial call, then the previous DQE address for each subsequent call

Exit Conditions

Return Sequence

M.RTRN 5

Registers

Normal Return:

R5	Bit 0 is set if more than one task satisfies the given parameters. Bits 1-31 contain the DQE address of the first matching task found. If no entry satisfies the given parameters, R5 equals zero. R5 may be used as input for subsequent calls.

Abnormal Return:

CC1 set	Invalid parameter block address. R5 remains unchanged.

6.2.66 M.INCLUDE - Include Memory Partition

The M.INCLUDE service allows a nonbase task to include dynamic or static memory partitions, or shared images into its address space. A static or dynamic partition can be included at the logical address specified at creation or at a logical address specified in the caller notification packet (CNP). The task is suspended until the inclusion is complete. If the resource was not included by another task, an allocated resource table (ART) and shared memory table (SMT) entry is established for the resource. The resource is automatically allocated for explicit shared use. If inclusion is successful, the assign and user counts are incremented for the resource.

The partition is identified by an 8-character partition name or a resource identifier (RID) that was defined when the partition was created. If the partition is dynamic, it is identified by an 8-character owner name that associates this copy of the partition with a set of users. The shared image is identified by a resource pathname or a resource identifier (RID) that was defined when the image was created.

Prezeroing of partitions is not performed by this service. The resource is swappable, if the user count goes to zero, and remains allocated until the assign count is zero.

The option is provided to lock the resource for exclusive use. The resource remains locked until: the owner of the lock terminates, the Release Exclusive Lock (M.UNLOCK) service is explicitly called, or the resource is excluded by the task.

If a partition has been included by a task, subsequent includes by that task are ignored.

The equivalent base mode service is M_INCLUDE.

Note: To ensure proper inclusion, the first eight characters of a shared image file name or partition name should be unique within the system.

Entry Conditions

Calling Sequence

M.INCLUDE [*addr*],[*task*] [,*cnpaddr*]

(or)

```
LW   R1,addr
LD   R4,task
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'40' (or) M.CALL H.REMM,12
```

addr is an address of a resource ID (RID), obtained when the partition was created

(or)

an address of a pathname vector containing the character count in byte 0 and the address of the pathname string in bytes 1, 2, and 3

task applies to dynamic partitions only; when supplied, specifies a doubleword address containing the left-justified task number of the original owner of the partition in word 0; word 1 is zero

(or)

a 1- to 8-character left-justified, blank-filled owner name

cnpaddr is the address of a caller notification packet (CNP) if notification is desired or if a logical address for partition inclusion is specified

Applicable portions of the CNP for this function are time-out value, abnormal return address, options field, and status field.

The option field has the following significance:

Bit	Meaning if Set
0	read/write access requested
1	reserved
2	set exclusive resource lock
3	reserved for MPX-32
4	logical address of partition is supplied in word 4
5-15	reserved

Defaults: If a CNP is not supplied, the partition is included as read-only with no lock established and at the logical address specified at creation.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R3,R5	M.RTRN R3,R5
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R3 contains the starting logical address of the shared memory partition

R5 contains the allocation index, a unique 32-bit integer number that may be used to set and release exclusive or synchronous locks on the partition while it is allocated. It contains the nonzero biased SMT index in the first byte and the address of the associated ART entry in the next three.

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
1	memory partition definition not found
2	specified access mode not allowed
8	unrecoverable I/O error to volume
10	dynamic partition definition exceeds one megabyte
16	memory requirements conflict with task's address space
38	time out occurred while waiting for shared memory to become available
50	partition is exclusively locked
55	allocated resource table (ART) is full
58	shared memory table (SMT) space unavailable
62	resource specified is not a static or dynamic partition and the option field indicates a logical inclusion address is given
80	shared image version level is not compatible with executable image
98	requires more shadow memory than exists

Wait Conditions

When the partition is not available (status values 50 to 58), the task is placed in a wait state, if specified in the CNP.

6.2.67 M.INQUIRY - Resource Inquiry

The M.INQUIRY service obtains information specific to a resource allocated by a nonbase mode task. The information is returned as pointers to the data structures that describe the resource. The resource must have been previously allocated, included for memory partitions, by the caller. Resources are identified by a logical file code obtained when the resource is allocated, a memory partition name defined when the partition is created, or an allocation index obtained when the resource is allocated or included. If not supplied as an argument, the caller is provided with the unique allocation index that can set and release exclusive or synchronous locks on the resource while it remains allocated. The caller must interpret the information in the identified structures as the application dictates. This should be done by a user-supplied subroutine that acts as a common interface between application programs and this service. Resource inquiries are then less sensitive to changes in system structures.

The base mode equivalent service is M_INQUIRER.

Entry Conditions

Calling Sequence

M.INQUIRY [*addr1*],[*addr2*][,*cnpaddr*]

(or)

```
LA   R1,addr1
LD   R4,addr2
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'48'  (or)  M.CALL  H.REMM,27
```

addr1 is the address of an 8-word parameter description area where the pointers to the appropriate system structure entries corresponding to this resource are to be returned

addr2 is the address of a doubleword containing zero in byte 0 and a 1- to 3-character, left-justified, blank-filled LFC in bytes 1, 2, and 3 of word 0 with word 1 zero

(or)

is the address of a doubleword containing the address of a 1- to 8-character, left-justified, blank-filled memory partition name in word 0 and the left-justified task number or address of a 1- to 8-character left-justified, blank-filled owner name in word 1

(or)

is the address of a doubleword containing zero in word 0 and the allocation index obtained when the resource was assigned in word 1

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R5	M.RTRN R5
(or)	(or)
M.RTNA R5 (CC1 set)	M.RTRN R5,R7 (CC1 set)

Registers

R5 contains the allocation index if not supplied as an argument, or zero if resource is undefined

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

The interpretation of each word in the parameter description area and some of the pertinent information that can be extracted from each structure is as follows:

Word 0 - Allocated resource table (ART) address:
- number of tasks assigned to this resource
- number of tasks currently using resource
- exclusive lock owner (DQE index)
- synchronous lock owner (DQE index)
- current allocation usage mode
- current allocation access mode (implicit shared)
- shared relative EOF block number (implicit shared)
- shared relative EOM block number (implicit shared)

Word 1 - File assignment table (FAT) address:
- relative EOF block
- relative EOM block
- number of segments in file
- current segment number
- current access mode
- relative file block position
- volume number (unformatted media only)
- unformatted ID (unformatted media only)
- assigned access restrictions
- file attribute and status flags

Word 2 - Unit definition table (UDT) address:
- device type code
- logical channel number
- logical subchannel number
- physical channel number (if different from logical)
- physical subchannel number (if different from logical)
- sectors per block (disk/floppy)
- sectors per allocation unit (disk/floppy)
- sectors per track (disk/floppy)
- number of heads (disk/floppy)
- total number of allocation units (disk/floppy)
- sector size (disk/floppy)

characters per line (TTY/terminal)
lines per screen (TTY/terminal)
tab size (TTY/terminal)
tab settings (TTY/terminal)

Word 3 - Device type table (DTT) address:
number of controller entries for device
ASCII device mnemonic
device type code

Word 4 - Controller definition table (CDT) address:
controller I/O class
number of devices on controller
device type code
interrupt priority level
logical channel number
logical subchannel number of first device
address of interrupt handler
interrupt vector location
controller definition flags

Word 5 - Shared memory table (SMT) address. Applies only to memory partitions:
number of tasks queued to partition
starting map register number
memory type
starting page number (for static partition only)
total number of pages (for static partition only)
number of map image descriptors
map image descriptor list

Word 6 - File pointer table (FPT) address:
logical file code associated with resource

Word 7 - Mounted volume table (MVT) address. Applies only to volume resources:
volume name
current number of users of volume
volume definition flags
root directory resource ID
number of descriptors available on volume
number of allocation units available
volume access restrictions

Notes:

1. A value of zero returned in any word of the parameter description area implies the corresponding structure does not apply to the resource for which the inquiry was made. For example, only words 0 and 5 apply for memory partitions.
2. For volume resources, words 2 through 4 pertain to the device upon which the volume is mounted.
3. The MPX-32 Technical Manual, Volume I, contains a complete description of the various system structures.

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
5	shared memory table entry not found for partition
29	logical file code not assigned
30	invalid allocation index

6.2.68 M.INT - Activate Task Interrupt

The M.INT service allows the calling task to cause the previously declared break or task interrupt receiver routine of the specified task to be entered.

The base mode equivalent service is M_INT.

Entry Conditions

Calling Sequence

M.INT *task*

(or)

```
ZR   R6          } (or)  LD R6,taskname
LW   R7,taskno   }
SVC  1,X'6F'  (or)  M.CALL H.REXS,47
```

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 unchanged if R7 is zero. If R7 is not zero, bit 0 of R6 is one if the specified task was not set up to receive a pseudointerrupt; otherwise bit 0 is zero. Bits 1-31 of R6 are zero in all cases.

R7 contains the task number, or zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted by M.KEY file from access to tasks with a different owner name

6.2.69 M.IPUBS - Set IPU Bias

The M.IPUBS service allows the user to dynamically change the IPU bias state for the current task.

The base mode equivalent service is M_IPUBS.

Entry Conditions

Calling Sequence

M.IPUBS *bias*

(or)

```
LI    R7,bias
SVC   2,X'5B'  (or)  M.CALL H.REXS,82
```

bias is the IPU bias state requested as follows:

Value	Description
0	nonbiased task; for example, can be executed by either the CPU or IPU
1	CPU only; for example, can be executed only by the CPU
2	IPU bias; for example, can be executed by either the CPU or IPU but is given priority status by the IPU

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6 contains execution status as follows:

Value	Description
0	normal return
1	IPU is not configured in the system
2	IPU is currently marked off-line

R7 IPU bias state of the task before this service was issued as follows:

Value	Description
0	nonbiased task
1	CPU only
2	IPU bias

6.2.70 M.LOC - Read Descriptor

The M.LOC service reads a resource descriptor for a specified resource. This service can examine the attributes of any volume resource. It is the responsibility of the caller to be familiar with the fields of the resource descriptor to determine the recorded information. It is recommended that this service is called by a user-supplied subroutine(s) which acts as a common interface between application programs and this service. Application programs are then less sensitive to changes in organization and content of these data structures.

The base mode equivalent service is M_READD.

Entry Conditions

Calling Sequence

M.LOC [*addr*],[*rdaddr*][,*cnpaddr*]

(or)

```
LW   R1,addr
LA   R6,rdaddr
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'2C' (or) M.CALL H.VOMM,13
```

addr	contains a pathname (PN) vector, a pathname block (PNB) vector, a resource identifier (RID) vector, an LFC, or a file control block (FCB) address
rdaddr	is the address of a read descriptor RD buffer. The buffer must be doubleword bounded and 192W length
cnpaddr	is the address of a CNP zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R4	M.RTRN R4
(or)	(or)
M.RTNA R2 (CC1 set)	M.RTRN R7,R2 (CC1 set)

Registers

Normal Return:

R4 contains the address of the mounted volume table entry (MVTE) for the specified volume

R7 contains the return status if a CNP is not supplied. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

Abnormal Return:

R2 contains the address of the last PN item processed

6.2.71 M.LOCK - Set Exclusive Resource Lock

The M.LOCK service allows a task to obtain exclusive allocation of a resource, as though it were nonshareable, for as long as the lock is owned. The resource must have been previously allocated or included for memory partitions. The resource is identified by either a logical file code (defined when the resource was assigned) or an allocation index obtained when the resource was assigned or by a resource inquiry. The task may request immediate denial if the lock is not available, or wait for an indefinite or specified period of time. An exclusive resource lock may be obtained for any allocated resource that is not being shared by multiple tasks at the time of the call to this service.

The base mode equivalent service is M_LOCK.

Entry Conditions

Calling Sequence

M.LOCK [*addr*][,*cnpaddr*]

(or)

```
LW   R5,addr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'44'  (or)  M.CALL  H.REMM,23
```

addr is the address of the allocation index obtained when the resource was assigned

(or)

is the address of a file control block (FCB) which contains a LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC is not assigned
30	invalid allocation index
38	time out occurred while waiting to become lock owner
46	unable to obtain resource descriptor lock (multiprocessor only)
50	resource is locked by another task
51	resource is allocated to another task

Wait Conditions

The task is placed in a wait state, as appropriate, if specified with the CNP.

6.2.72 M.LOGR - Log Resource or Directory

The M.LOGR service provides the nonbase mode task with a convenient interface to locate the directory entry and resource descriptor for a single resource or for all the resources defined in a specified directory.

The caller must make a resource specification in the resource logging block (RLB). The log service will evaluate the resource specification and determine whether to log a single resource or all the resources defined in a directory. Some resource specifications are ambiguous and require the caller to specify additional information so the type of log function requested can be determined.

To log all the resources defined in a specified directory, the M.LOGR service must be called repeatedly until the last resource in the directory is logged. The user must reset bit 0 to zero in RLB.TYPE to indicate the first call. The operating system automatically changes the contents of bit 0 to one to indicate recall. Once all resources on the directory are logged, the operating system automatically resets bit 0 back to zero to indicate all resources have been logged.

Note: The M.LOGR system service does not search the memory resource descriptor table (MDT) for resource descriptors.

6.2.72.1 Resource Specifications for Pathnames

The caller can specify any valid pathname that is recognized by the Volume Management Module. The log service recognizes all valid pathname variations. However, some pathnames are ambiguous within the context of this service and require special considerations for the service to function with the expected results.

Specifically, pathnames that end with a directory specification are interpreted to mean log the contents of the directory. Directories can be logged as resources in two ways. The first is to supply a pathname that specifies the directory as a resource. This specification is not ambiguous. The second way is to supply a pathname that ends with a directory specification. This type of pathname is ambiguous and requires special handling.

Examples

The following type of pathname always logs the directory entry and resource descriptor for the specified resource.

@volume (directory) resource

The following type of pathname usually specifies to log the contents of the specified directory. The meaning of this pathname can be changed by setting the log single flag (RLB.LS) bit in the RLB flag word (RLB.INT). When the RLB.LS flag is set, the directory entry and resource descriptor for the specified directory are returned.

@volume (directory)

The following type of pathname means log the specified directory. The directory entry and resource descriptor for the specified directory are returned.

@volume^directory

6.2.72.2 Resource Specifications for Pathname Blocks

Pathname blocks are processed in the same manner as pathnames.

6.2.72.3 Resource Specifications for a Resource Identifier

When a resource identifier (RID) is furnished, the log service assumes the indicated resource is a directory and attempts to log the indicated resource as a directory.

6.2.72.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index

When this type of resource specification is provided the log service makes the following assumptions:

- The implied file control block (FCB) is assigned to a directory.
- The implied FCB is opened.
- The buffer address in the FCB is the buffer to be used by the log service for locating directory entries.
- The transfer quantity in the FCB is the maximum size of the directory entry buffer.
- The FCB must be an extended FCB and must be opened in random access mode.
- The buffer is empty on the initial call and positions to the beginning of the directory and primes the supplied buffer. The directory is not read again until it is exhausted.

The caller should assign the directory in read mode so the directory can be searched by other users as it is being logged.

The base mode equivalent service is M_LOGR.

Entry Conditions

Calling Sequence

M.LOGR [*rlbaddr*][,*cnpaddr*]

(or)

```
LA   R2,rlbaddr
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'29' (or) M.CALL H.VOMM,10
```

rlbaddr	is the address of the resource logging block (RLB)
cnpaddr	is a CNP address or zero if not supplied

RLB Structure on Initial Call

	0 7	8 15	16 23	24 31
Word 0	PN vector or RID vector or zero (RLB.TGT). See Note 1.			
1	192-word buffer address or zero (RLB.BUFA). See Note 2.			
2-3	Reserved for system use			
3	Parent directory RD block address (RLB.RDAD)			
4	Type (RLB.TYPE) See Note 3.	Zero (RLB.BOFF). See Note 3.		
5	Length. See Note 4.	Directory return buffer address (RLB.DIRA). See Note 4.		
6	User FCB address or zero (RLB.FCB). See Note 5.			
7	Flags. See Note 6.	Reserved (RLB.INT)		

Notes:

1. If the PN vector length and address specify a resource, only one item is logged. If the specification does not end with a resource, but with a directory, the entire directory may be logged by repeated calls. A call by RID vector implies the RID is for a directory and all entries may be logged. A value of zero implies the entire contents of the current working directory.
2. This address must be doubleword bounded if this field is zero, the RD is not returned.
3. The type value should be zero if the call is by PN vector (length and address) or zero to indicate working directory. Type should be one to indicate a call by RID. If all resources in a directory are to be logged, bit 0 of RLB.TYPE must be zero to indicate the first call.
4. This word contains the address of a buffer and its length in words. The buffer may be up to 16 words long. The log service will place the first *n* words of the logged directory entry into this buffer. This provides the user access to the file name and other attributes that exist only in the directory entry.
5. This service uses the system FCB by default. Phasing problems may occur, as the directory to be logged must be deassigned between calls if multiple entries are desired. In many cases, the impact of having an entry deleted just after it has been logged, or having an entry appear after that portion in the directory has been scanned, will be small or nonexistent. In other cases, such as saving files in a directory, it may be major. To prevent these problems, the address of a FCB that will be used to hold the directory while logging occurs may be provided.

6. Bits in this word are assigned as follows:

Bit	Description
0-1	reserved
2	if set, directory entry and resource descriptor for specified directory are returned (RLB.LS)
3	root directory
4	used on return to indicate whether resource was located (see description of RLB Structure on Return under Exit Conditions)
5-7	reserved

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTNA (CC1 set)	M.RTRN R7

Registers

R7 return status if a CNP is not supplied; otherwise CNP address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

RLB Structure on Return

	0 7	8 15	16 23	24 31
Word 0	PN vector or RID vector or zero (RLB.TGT)			
1	192-word buffer address or zero (RLB.BUFA)			
2	MVTE address (RLB.MVTE). See Note 1.			
3	Disk address of directory RD (RLB.RDAD). See Note 1.			
4	Type (RLB.TYPE) See Note 2.	Byte offset of entry (RLB.BOFF).		
5	Length	Directory return buffer address (RLB.DIRA)		
6	User FCB address or zero (RLB.FCB)			
7	Flags. See Note 3.	Reserved (RLB.INT).		

Notes:

1. When all resources in a directory are to be logged, RLB.MVTE and RLB.RDAD are used by the operating system as input after the first call.

2. The operating system automatically changes the contents of bit 0 in RLB.TYPE as follows:

Value	Description
0	all resources in the directory have been logged; do not recall this service
1	recall this service and log the next resource in the directory

3. Bits in this word are assigned as follows:

Bit	Contents
0-1	reserved
2	directory entry and resource descriptor for specified directory are returned
3	root directory
4	zero if resource was not located
5-7	reserved

6.2.73 M.MEM - Create Memory Partition

The M.MEM service creates permanent memory partition definitions. Permanent memory partition definitions are given names in directories and remain known to the operating system until explicitly deleted.

MPX-32 uses memory partition definitions to relate named globally accessible areas of memory to the tasks that require them.

This service allocates a resource descriptor and defines the memory requirements for the partition. Next, the attributes of the partition are recorded in the resource descriptor. Finally, the name of the partition is established in the indicated directory.

When a directory entry is established, the directory entry is linked to the resource descriptor for the partition. This link relates the name of the partition to the other attributes of the partition. Typical partition attributes are:

- partition name
- partition resource identifier (RID)
- partition protection attributes
- partition management attributes
- partition memory requirements

Asynchronous abort and delete are inhibited for execution of this service.

The base mode equivalent service is M_MEM.

Entry Conditions

Calling Sequence

M.MEM *vector,rcbaddr*[,*cnpaddr*]

(or)

```
LW   R1,vector
LA   R2,rcbaddr
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'22' (or) M.CALL H.VOMM,3
```

vector	contains a PN vector or PNB vector
rcbaddr	is the address of the RCB
cnpaddr	is the address a CNP or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.74 M.MEMB - Get Memory in Byte Increments

The M.MEMB service allows task to dynamically expand its memory allocation in doubleword increments starting at the end of its DSECT up to the top of its logical address space. The additional memory is the same type specified when the task was cataloged. The task is mapped in a logically contiguous manner up to the end of its address space. The task is suspended until the allocation is successful. Repeated calls to this service are allowed. Allocation is not contiguous with previously allocated space.

This service cannot be used with the M.GE or M.GD services, or a call to H.MEMM,14.

The base mode equivalent service is M_GETMEMBYTES.

Entry Conditions

Calling Sequence

M.MEMB *bytes*

(or)

LW R4,=*bytes* (or) LI R4,*bytes*
SVC 2,X'4B' (or) M.CALL H.REMM,28

bytes is the number of bytes to allocate

Exit Conditions

Return Sequence

M.RTRN R3,R4

Registers

CC1	contains zero
CC2	contains zero
R3	contains the 24-bit starting logical doubleword address of allocated space
R4	contains the number of bytes actually allocated (modulo 2W)

(or)

CC1	contains zero
CC2	contains zero
R3	contains the 24-bit starting logical doubleword address of allocated space
R4	contains the number of bytes actually allocated (modulo 2W); however, the number is less than requested.

Error Condition

Allocation Denied:

CC1	contains one
CC2	contains one
R3	contains zero
R4	contains zero

6.2.75 M.MEMFRE - Free Memory in Byte Increments

The M.MEMFRE service allows a task to dynamically deallocate acquired memory. Deallocation can be random. The space address must have been previously obtained from the M.MEMB service. All of the space obtained from a given call is deallocated.

This service cannot be used with the M.FE or M.FD services.

The base mode equivalent service is M_FREEMEMBYTES.

Entry Conditions

Calling Sequence

M.MEMFRE *addr*

(or)

```
LW   R3,addr
SVC  2,X'4C'  (or)  M.CALL   H.REMM,29
```

addr is the starting address of a dynamic space previously acquired from the M.MEMB service

Exit Conditions

Return Sequence

M.RTRN R3 (or) abort user with RM77

Registers

R3 contains zero if deallocation could not be performed. Deallocation address was not found in allocation table

Abort Cases

RM77 A TASK HAS DESTROYED THE ALLOCATION LINKAGES IN ITS DYNAMIC EXPANSION SPACE

6.2.76 M.MOD - Modify Descriptor

The M.MOD service allows the owner of a resource to change the protection or other resource management attributes of a resource. The owner can restrict or allow attributes using this service.

Only certain information in a descriptor can be changed.

The caller is allowed to modify the following:

- the protection fields of the descriptor
- the accounting fields of the descriptor
- the extension attribute fields of the descriptor
- words 160 through 175 of the user data field of the descriptor
- the shared image field of the descriptor

Information such as the volume space occupied by the resource cannot be changed as this would allow the caller to violate the integrity of the volume on which the resource resides.

This service is the first part of a two step operation. The caller is required to read the resource descriptor into memory in order to modify it. Once in memory, the resource descriptor is locked until the caller writes the modified descriptor back to the volume using the Rewrite Descriptor (M.REWRIT) service. The caller must issue the rewrite before modifying another descriptor.

Only the resource owner or the system administrator can modify a resource descriptor. The format of the descriptor and the type of data to be modified must be known by the modifier.

The base mode equivalent service is M_MOD.

Entry Conditions

Calling Sequence

M.MOD [*addr*],[*rdaddr*][,*cnpaddr*]

(or)

```
LW   R1,addr
LA   R6,rdaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'2A' (or) M.CALL  H.VOMM,11
```

addr	contains a PN vector, a PNB vector, or a RID vector
rdaddr	is the address of an RD buffer doubleword bounded and 192W length
cnpaddr	is the address of a CNP or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.77 M.MODU - Modify Descriptor User Area

The M.MODU service allows users with write, update, append, or modify access to a resource to change the user area of the resource descriptor of that resource.

The user can change all fields in the user area of the resource descriptor, however, some utilities use words 176 through 190. Only words 160 through 175 should be modified. Word 191 of the resource descriptor is a reserved location; any changes to this word are ignored.

This service is the first part of a two step operation. The caller is required to read the user area of the resource descriptor into memory in order to modify it. Once in memory, the resource descriptor is locked, for example, protected from access until the caller writes the modified user area back to the volume using the Rewrite Descriptor User Area (M.REWRTU) service. The caller must issue the rewrite before modifying another descriptor or descriptor user area.

The base mode equivalent service is M_MODU.

Entry Conditions

Calling Sequence

M.MODU [*addr*],[*uaaddr*][,*cnpaddr*]

(or)

```
LW   R1,addr
LA   R6,uaaddr
LA   R7,cnpaddr
SVC  2,X'31'  (or)  M.CALL H.VOMM,26
```

addr	contains a PN vector, a PNB vector, or an RID vector
uaaddr	is the address of a user area buffer doubleword bounded and 32W in length
cnpaddr	is the address of a CNP or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.78 M.MOUNT - Mount Volume

The M.MOUNT service establishes a task or TSM environment as a user of a volume. If the volume requested is not physically mounted, M.MOUNT notifies the operator to mount the volume, and creates a mounted volume table (MVT) entry for the volume. This entry remains in memory as long as there are established users of the volume.

If the volume requested is already physically mounted, M.MOUNT attempts a logical mount.

For nonpublic volumes, M.MOUNT allocates a volume assignment table (VAT) entry within the user's TSA, provided that the requested usage classification is compatible. A request to mount a public or nonpublic volume that is already physically and logically mounted is ignored.

The base mode equivalent service is M_MOUNT.

Entry Conditions

Calling Sequence

M.MOUNT [*rrsaddr*][,*cnpaddr*]

(or)

```
LA   R1,rrsaddr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'49'  (or)  M.CALL  H.REMM,17
```

rrsaddr is the address of a resource requirement summary (RRS) entry type 9

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
1	invalid volume name
2	volume use not allowed to this user
6	volume assignment table (VAT) space not available
8	unrecoverable I/O error to volume
11	invalid RRS entry
13	mount device not in system
18	invalid mount device specified
20	unable to initialize volume - volume unsafe
21	J.MOUNT run request failed
34	attempt to mount a public volume without the system administrator attribute
37	invalid J.MOUNT request
38	time out occurred while waiting for resource
39	volume already mounted
42	user requested abort of mount process
43	user requested hold on mount process
53	mount device unavailable
55	allocated resource table (ART) space not available for mount device
57	unable to mount volume for requested usage
59	mounted volume table (MVT) space not available
73	file overlap occurred
91	unable to mount volume due to pending physical dismount
93	unable to perform physical mount due to system shutdown in progress
94	J.MOUNT attempted to mount an unformatted disk volume

Abort Cases

```
RM42     USER REQUESTED ABORT OF MOUNT PROCESS
```

Wait Conditions

When the volume is unavailable as indicated by status values 50-63, the task is placed in a wait state, as appropriate.

6.2.79 M.MOVE - Move Data to User Address

The M.MOVE service is used to move data in the task's logical address space to areas of the task's logical address space which are write protected. This service cannot be used in shared read only memory partitions.

The base mode equivalent service is M_MOVE.

Entry Conditions

Calling Sequence

M.MOVE *inbuffer, outbuffer, number*

(or)

```
LA   R1,inbuffer
LA   R2,outbuffer
LI   R4,number
SVC  2,X'62' (or) M.CALL H.REXS,89
```

inbuffer	is the byte address of the buffer to be moved
outbuffer	is the destination byte address
number	is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Normal Return Sequence:

M.RTRN

Abnormal Return Sequence:

M.RTRN R7

Registers

CC1 set error condition
R7 contains error condition as a decimal value as follows:

Value	Description
256	invalid source buffer address
257	destination buffer is in the operating system
258	destination buffer is in the TSA
259	invalid destination buffer address
261	invalid number of bytes to be moved

6.2.80 M.MYID - Get Task Number

The M.MYID service allows the user to obtain status on the currently executing task.

The base mode equivalent service is M_MYID.

Entry Conditions

Calling Sequence

M.MYID *pbaddr*

(or)

```
ZR   R5
SBR  R5,0
LA   R7,pbaddr
SVC  1,X'64' (or)  M.CALL  H.REXS,32
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

Word	Contents
0	task activation sequence number
1-2	task load module name
3-4	owner name
5-6	pseudonym
7-8	current working directory, truncated to the first eight characters
9	reserved
10	scheduling flags used by the scheduler to indicate special status conditions

Exit Conditions

Return Sequence

M.RTRN 5

Registers

CC1 set	invalid parameter block address
R5,R7	unchanged

Abort Cases

RX32	INVALID DQE ADDRESS

6.2.81 M.NEWRRS - Reformat RRS Entry

The M.NEWRRS service converts a resource requirement summary (RRS) entry from MPX-32 Revision 1.*x* format to the format acceptable for assignment processing by the Resource Management Module (H.REMM). See the MPX-32 Revision 1.*x* Technical Manual. It is intended for compatibility purposes and should only be used when internal recoding of the RRS entry to the new format is impractical. This service is automatically called during parameter task activation when an incompatible RRS format is encountered in the activation parameter block. This results in additional overhead during the activation of these tasks.

Entry Conditions

Calling Sequence

M.NEWRRS *rrsaddr,newaddr*[,*cnpaddr*]

(or)

```
LA   R1,rrsaddr
LA   R5,newaddr
LA   R7,cnpaddr  (or) ZR  R7
SVC  2,X'54'  (or)  M.CALL H.REXS,76
```

rrsaddr is the address of the RRS entry to be reformatted

newaddr is the address where the reformatted RRS entry is to be built

Restrictions:

This area where the reformatted RRS entry is to be built must be large enough to account for the expanded size of the new RRS. Use the following guidelines to determine the number of words required to accommodate the expansion:

MPX-32 1.*x* RRS type	Size if reformatted RRS
ASSIGN1	12 words
ASSIGN2	4 words
ASSIGN3 (disk)	4-8 words
ASSIGN3 (device)	6 words
ASSIGN4	4 words

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Registers

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
11	invalid RRS entry
20	specified volume not in system

6.2.82 M.OLAY - Load Overlay Segment

The M.OLAY service provides for loading an overlay segment into a nonbase mode task. The actual loading is done by the Task Management Module (H.TAMM) and control is returned to the caller upon completion, or to the overlay segment at its cataloged transfer address. The named segment must have been defined to the Cataloger as an overlay.

Entry Conditions

Calling Sequence

M.OLAY *filename*[,EXE]

(or)

```
LD    R6,filename
SVC   1,X'50'  (or)  M.CALL H.REXS,13   } (or)
SVC   1,X'51'  (or)  M.CALL H.REXS,14   }
```

filename is a doubleword containing the name of the file from which the overlay segment is to be loaded. The name must be 1 to 8 ASCII characters, left-justified and blank-filled

EXE specifies transfer control to the overlay (SVC 1,X'51'). If not specified, control returns to the caller.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the transfer address of the overlay segment or unchanged if the EXE parameter is specified

Abort Cases

RX07, LD01-LD08
These abort messages occur when M.OLAY cannot load the overlay segment due to software checksum or data error.

RX08 OVERLAY IS NOT IN THE DIRECTORY

RX10 OVERLAY HAS AN INVALID PREAMBLE

RX11 AN UNRECOVERABLE I/O ERROR HAS OCCURRED DURING OVERLAY LOADING

RX33 OVERLAY LINKAGES HAVE BEEN DESTROYED BY LOADING A LARGER OVERLAY

6.2.83 M.OPENR - Open Resource

The M.OPENR service prepares a resource for logical I/O and defines the intended access mode for subsequent operations on the resource. Protection is provided for both the requestor and the resource against indiscriminate access. If appropriate, additional FAT information is posted at this time. A blocking buffer can be allocated if not previously specified, explicitly or implicitly, during allocation of the resource. However, if a user-supplied buffer is specified in the FCB, that buffer is used and any previously allocated blocking buffer is released. A mount message is issued as a result of this function when the I/O is to be performed to a device associated with unformatted media, and the message has not been inhibited by user request or previous open on the resource by another user. An open request to a resource that is already opened in the same access mode is ignored.

The base mode equivalent service is M_OPENR.

Entry Conditions

Calling Sequence

M.OPENR *fcbaddr* [,*cnpaddr*]

(or)

```
LA   R1,fcbaddr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'42'  (or)  M.CALL  H.REMM,21
```

fcbaddr is the address of a file control block (FCB)

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are time-out value, abnormal return address, options field, and status field.

The options field describes the access and usage specification with the following interpretation:

Byte 0 contains an integer value representing the specific access for which the resource is being opened. An error message will be returned for an invalid integer. The following values are valid:

Value	Description
0	open for default access
1	open for read
2	open for write (resource redefined)
3	open for modify
4	open for update
5	open for append
6-255	reserved

Byte 1 indicates the usage under which the resource is to be opened with the following significance:

Bit	Meaning if Set
0	open for explicit shared use
1	open for exclusive use
2	open in unblocked mode overrides any specification made at resource assignment
3	open in blocked mode overrides any specification made at resource assignment
4	resource data blocked. If the file is actually written to in any access mode (append, modify, update, write), the data will be recorded as blocked in the resource descriptor when the file is closed, regardless of whether the I/O was actually performed in blocked mode.
5	override with implicit shared use
6-7	reserved

Only one of bits 0 and 1 in byte 1 can be set. If set, any usage specified at the time the resource was assigned is overridden. This can result in a denial condition if the usage specified at open differs from that specified at assignment.

If a CNP is not supplied or the specification in the options field is zero: the resource is opened for read access for a volume resource or update access for a device unless only a specific access mode was allowed at assignment, in which case the resource is opened for that access; the usage is implicit shared or that specified at resource allocation, whichever is appropriate.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.84 M.OSREAD - Physical Memory Read

The M.OSREAD service moves an arbitrary number of bytes from a location in the physical space of MPX-32 to a location in the calling task's logical address space.

The physical space of MPX-32 includes:

- nonextended MPX-32 — location 0 through the end of memory pool
- extended MPX-32 — the beginning of extended MPX-32 through the end of the last map block of extended MPX-32

The base mode equivalent service is M_OSREAD. This service is executable by the IPU.

Entry Conditions

Calling Sequence

M.OSREAD [*inbuffer*] [,*outbuffer*] [,*number*]

(or)

```
LA   R1,inbuffer
LA   R2,outbuffer
LI   R4,number
SVC  2,X'7E'     (or)   M.CALL   H.REXS,93
```

inbuffer is the 24-bit pure address of the byte bounded source buffer (in the physical space of MPX-32)

outbuffer is the 24-bit pure address of the byte bounded destination buffer (in the task's logical address space)

number is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Normal Return:

M.IPURTN

Abnormal Return:

M.IPURTN R7

Registers

CC1 set error condition

R7 error condition as a decimal value:

Value	Description
256	invalid source buffer address
259	invalid destination buffer address
261	invalid number of bytes to be moved

6.2.85 M.OSWRIT - Physical Memory Write

The M.OSWRIT service moves an arbitrary number of bytes from a location in the logical address space of the calling task to a location in the physical space of MPX-32.

The physical space of MPX-32 includes:

- nonextended MPX-32 — location 0 through the end of memory pool
- extended MPX-32 — the beginning of extended MPX-32 through the end of the last map block of extended MPX-32

This service is available only to privileged users and is executable by the IPU. The base mode equivalent service is M_OSWRIT.

Entry Conditions

Calling Sequence

M.OSWRIT [*inbuffer*] [,*outbuffer*] [,*number*]

(or)

```
LA   R1,inbuffer
LA   R2,outbuffer
LI   R4,number
SVC  2,X'AF'   (or)   M.CALL   H.REXS,94
```

inbuffer is the 24-bit pure address of the byte bounded source buffer (in the task's logical address space)

outbuffer is the 24-bit pure address of the byte bounded destination buffer (in the physical space of MPX-32)

number is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Normal Return:

M.IPURTN

Abnormal Return:

M.IPURTN R7

Registers

CC1 set error condition
R7 error condition as a decimal value:

Value	Description
256	invalid source buffer address
259	invalid destination buffer address
261	invalid number of bytes to be moved

6.2.86 M.PGOD - Task Option Doubleword Inquiry

The M.PGOD service provides the caller access to both the first and second program option words. The first option word, which is the same word as supplied by the M.PGOW service, resides in R7. The second word resides in R6.

This service can be executed by the IPU.

The base mode equivalent service is M_OPTIONDWORD.

Entry Conditions

Calling Sequence

M.PGOD

(or)

SVC 2,X'C0' (or) M.CALL H.REXS,95

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6 contains the 32-bit second option word
R7 contains the 32-bit first option word

6.2.87 M.PGOW - Task Option Word Inquiry

The M.PGOW service provides the caller with the first 32-bit task option word. This word is also called the program option word. Use the M.PGOD service to return the first and second task option words.

This service can be executed by the IPU.

The base mode equivalent service is M_OPTIONWORD.

Entry Conditions

Calling Sequence

M.PGOW

(or)

SVC 1,X'4C' (or) M.CALL H.REXS,24

Exit Conditions

Return Sequence

M.IPURTN 7

Registers

R7 contains the 32-bit first option word

6.2.88 M.PNAM - Reconstruct Pathname

The M.PNAM service constructs and returns the pathname string that was used to assign a file. In most cases, this service acquires the complete name of a file that was statically assigned to a task. If a pathname component contains special characters, the component is returned enclosed within single quotes.

The base mode equivalent service is M_CONSTRUCTPATH.

Entry Conditions

Calling Sequence

M.PNAM *arga,pnaddr*[,*cnpaddr*]

(or)

```
LW   R1,arga
LW   R4,pnaddr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'2F'  (or)  M.CALL  H.VOMM,16
```

arga	is an FCB address or LFC for the assigned volume resource
pnaddr	is the PN address and maximum pathname length
cnpaddr	is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R4	M.RTRN R4
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R4	contains the actual PN length and PN address
R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.89 M.PNAMB - Convert Pathname to Pathname Block

The M.PNAMB service converts a pathname to a form that can be more easily analyzed by software. In most cases, utility programs use this to syntax check a pathname that is read from a directive line.

When called, this service parses the entered pathname. If errors are detected in the pathname syntax, this service is terminated and R1 is updated to indicate the point where the error was detected.

The base mode equivalent service is M_PNAMB.

Entry Conditions

Calling Sequence

M.PNAMB *pnaddr,pnbaddr*[,*cnpaddr*]

(or)

```
LW   R1,pnaddr
LW   R4,pnbaddr
LA   R7,cnpaddr   (or) ZR   R7
SVC  2,X'2E'      (or) M.CALL  H.VOMM,15
```

pnaddr is the address and length of a PN vector, or zero for the current working volume and directory

pnbaddr is the address and length of a PNB vector, or zero for the current working volume and directory

cnpaddr is a CNP address or zero if CNP not supplied

Applicable portions of the CNP for this function are time-out value, abnormal return address, option field, and status field. The options field of the CNP has the following interpretation:

Byte 0 is reserved

Byte 1 has the following significance when set:

Bit	Meaning if Set
0-6	reserved
7	parsing of the pathname includes only the volume and directory portions of the supplied pathname. This bit is usually set by J.TSM.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R1,R4	M.RTRN R1,R4
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R1	contains the address of first PN character not processed and remaining length
R4	contains the PNB address and the actual PNB length
R7	contains the return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.90 M.PRIL - Change Priority Level

The M.PRIL service dynamically alters the priority level of the specified task. The caller must be privileged. Valid priority levels for real-time tasks are 1-54 inclusive. Valid priority levels for time distribution tasks are 55-64 inclusive. A real-time task cannot be changed to a time distribution priority level and a time distribution task cannot be changed to a real-time priority level. I/O continues to operate at base priority level of the cataloged task.

The base mode equivalent service is M_PRIL.

Entry Conditions

Calling Sequence

M.PRIL *task,priority*

(or)

```
LW   R5,priority
ZR   R6            } (or) LD  R6,taskname
LW   R7,taskno     }
SVC  1,X'4A'  (or) M.CALL  H.REXS,9
```

task is the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

priority is the priority level to be assigned to the task. Valid priorities are 1 through 54 for a real-time task and 55 through 64 for a time-distribution task.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the task number or zero if the specified task was not found

Abort Cases

RX06 UNPRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY LEVEL, OR A PRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY TO A LEVEL OUTSIDE THE RANGE OF 1 TO 64, INCLUSIVELY

6.2.91 M.PRIV - Reinstate Privilege Mode to Privilege Task

The M.PRIV service returns a task that was cataloged as privileged to the privileged status if the privilege was changed with the M.UPRIV service.

The base mode equivalent service is M_PRIVMODE.

Entry Conditions

Calling Sequence

M.PRIV

(or)

SVC 2,X'57' (or) M.CALL H.REXS,78

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set successful operation

6.2.92 M.PTSK - Parameter Task Activation

The M.PTSK service overrides specific task parameters in the load module or executable image preamble during activation. For unprivileged callers, some parameters are overridden by those of the calling task. The task name, optional resource requirements, and optional pseudonym are specified to the service call. When a task name is supplied in words 2 and 3 of the parameter task activation block (PTASK), the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied.

Options 1-32 reside in task option word 1. Options 33-64 reside in task option word 2. If using options 33-64, the expanded PTASK block format must be used and bit 4 of PTA.FLG2 must be set.

The base mode equivalent service is M_PTSK.

Entry Conditions

Calling Sequence

M.PTSK *actaddr*[,*psbaddr*]

(or)

```
LA   R1,actaddr
LA   R2,psbaddr (or) ZR R2
SVC  1,X'5F'  (or) M.CALL H.REXS,40
```

actaddr is the logical word address of the parameter task activation block. Words 0 to 12 of the block are required if options 1 through 32 are set; variable length RRS entries beginning in word 13 are optional. Words 0 to 19 are required if options 33 through 64 are set and bit 4 of PTA.FLG2 is set; variable length RRS entries beginning in word 20 are optional. The structure of the PTASK block follows.

psbaddr is the logical address of the parameter send block (PSB) or zero if no parameters are to be passed. If a load module name is supplied in the PSB, the load module must be in the system directory. A pathname vector or RID vector must be supplied if a load module is to be activated from a user directory.

The following is the structure of the expanded parameter task activation block:

Byte	Word	0 7	8 15	16 23	24 31
0	0	PTA.FLAG	PTA.NRRS	PTA.ALLO	PTA.MEMS
4	1	PTA.NBUF	PTA.NFIL	PTA.PRIO	PTA.SEGS
8	2-3	PTA.NAME			
10	4-5	PTA.PSN			
18	6-7	PTA.ON			
20	8-9	PTA.PROJ			
28	10	PTA.VAT	PTA.FLG2	PTA.EXTD	
2C	11	PTA.PGOW			
30	12	PTA.TSW			
34	13	PTA.RPTR			
38	14	PTA.PGO2			
3C	15	PTA.FSIZ		PTA.RSIZ	
40	16-19	Reserved (zero)			
50-*nn*	20-*nn*	RRS List			

Byte (Hex)	Symbol	Description
0	PTA.FLAG	contains the following:

Bit	Contents
0	reserved
1	job oriented (PTA.JOB)
2	terminal task (PTA.TERM)
3	batch task (PTA.BTCH)
4	debug overlay required (PTA.DOLY)
5	resident (PTA.RESD)
6	directive file active (PTA.DFIL)
7	SLO assigned to SYC (PTA.SLO)

For unprivileged callers, bits 0-3 are not applicable. These characteristics are inherited from the parent task.

1	PTA.NRRS	number of resource requirements or zero if same as summary entries in the load module or executable image preamble

Byte (Hex)	Symbol	Description
2	PTA.ALLO	memory requirement: number of 512-word pages exclusive of TSA, or zero if same as the preamble
3	PTA.MEMS	memory class (ASCII E, H or S) or zero if memory class is to be taken from the preamble. If the memory class is to be taken from the preamble, the caller has the option of specifying the task's logical address space in this field as follows: Bits — Contents 0-3 — hexadecimal value 0 through F representing the task's logical address space in megabytes where zero is 1MB and F is 16MB 4-7 — zero
4	PTA.NBUF	the number of blocking buffers required or zero if same as the preamble
5	PTA.NFIL	the number of FAT/FPT pairs to be reserved or zero if same as the preamble
6	PTA.PRIO	the priority level at which the task is to be activated or zero for the cataloged load module priority. See the Parameter Send Block section in Chapter 2 of this manual for more details.
7	PTA.SEGS	the segment definition count or reserved (zero)
8	PTA.NAME	contains the load module or executable image name, left justified and blank filled, or word 2 is zero and word 3 contains a pathname vector or RID vector
10	PTA.PSN	contains the 1- to 8-character ASCII pseudonym, left justified and blank filled, to be associated with the task or zero if no pseudonym is desired. For unprivileged callers, this attribute is inherited from the parent task if zero is supplied or the parent is in a terminal or batch job environment.
18	PTA.ON	contains the 1- to 8-character ASCII owner name, left-justified and blank-filled, to be associated with the task or zero if the task to default to the current owner name. Valid only when task has system administrator attribute.
20	PTA.PROJ	contains the 1- to 8-character ASCII project name, left-justified and blank-filled, to be associated with files referenced by this task, or zero if same as LMIT
28	PTA.VAT	the number of volume assignment table (VAT) entries to reserve for dynamic mount requests or zero if same as the preamble

Byte (Hex)	Symbol	Description
29	PTA.FLG2	contains the following flags: Bit — Meaning if Set 0 — debug activating task (PTA.DBUG) 1 — Command Line Recall and Edit is in effect for the task (PTA.CLRE) 2 — NOTSA option (PTA.NTSA) 3 — TSA option (PTA.TSA) 4 — expanded PTASK block flag (must be set to use options 33-64) (PTA.EBLK) 5 — reserved 6 — NOMAPOUT option (PTA.NMAP) 7 — MAPOUT option (PTA.MAP)
2A	PTA.EXTD	contains the following values: Bit — Meaning if Set -1 — maxaddr of extended MPX-32 and TSA -2 — minaddr of extended MPX-32 and TSA 0 — invalid with PTA.TSA or PTA.NTSA option *n* — a positive number representing the starting map block of extended MPX-32 and TSA
2C	PTA.PGOW	contains the initial value of the task option word or zero
30	PTA.TSW	contains the initial value of the task status word or zero
34	PTA.RPTR	contains a pointer to the resource requirement summary list or, if an expanded PTASK block is not used, the RRS list begins here (see RRS list description - byte 50)
38	PTA.PGO2	contains the initial value of the second task option word
3C	PTA.FSIZ	contains the length of the fixed portion of the PTASK block in bytes
3E	PTA.RSIZ	contains the number of bytes of the resource requirement summary
40	Reserved	
50		resource requirement summary list. Each entry contains a variable length RRS. The RRS list has up to 384 words. Each entry must be doubleword bounded. Each entry is compared with the RRS entries in the LMIT. If the logical file code currently exists, the specified LFC assignment will override the cataloged assignment, otherwise the special assignment will be treated as an additional requirement and merged into the list. If MPX-32 Revision 1.*x* format

of the RRS is specified, it is converted to the format acceptable for assignment processing by the Resource Management Module (H.REMM). See MPX-32 Revision 1.*x* Technical Manual for format of the RRS.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 equals zero if the service was performed

R7 contains the task number of the task activated by this service

(or)

R6 equals one if invalid attempt to multicopy a unique task

R7 task number of existing task with same name

(or)

R0 destroyed

R6

Value	Description
2	file specified in words 2 and 3 of the PTASK block not in directory
3	unable to allocate file specified in words 2 and 3 of the PTASK block
4	file is not a valid load module or executable image
5	DQE is not available
6	read error on resource descriptor
7	read error on load module
8	insufficient logical/physical address space for task activation
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task receiver queue full
16	invalid PSB address
17	RRS list exceeds 384 words
18	invalid RRS entry in parameter block

R7 contains zero if task not found

6.2.93 M.QATIM - Acquire Current Date/Time in ASCII Format

The M.QATIM service acquires the system date and time in ASCII format. The date and time are returned in a four word buffer, the address of which is specified in the call. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

The base mode equivalent service is M_QATIM

Entry Conditions

Calling Sequence

M.QATIM *addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'03000000'
SVC     2,X'50'   (or)   M.CALL H.REXS,74
```

addr is the address of a 4-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

```
RX13    FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS
        OUT OF RANGE
```

6.2.94 M.RADDR - Get Real Physical Address

The M.RADDR service allows unprivileged tasks to determine the physical memory address associated with a given logical address.

The base mode equivalent service is M_RADDR.

Entry Conditions

Calling Sequence

M.RADDR [*logicaladdr*]

(or)

```
LA   R1,logicaladdr
SVC  1,X'0E'   (or)   M.CALL H.REXS,90
```

logicaladdr is the logical address to be translated

Exit Conditions

Return Sequence

Registers

R7 contains the physical address

6.2.95 M.RCVR - Receive Message Link Address

The M.RCVR service allows the caller to establish the address of a routine to be entered for receiving messages sent by other tasks.

The base mode equivalent service is M_RCVR.

Entry Conditions

Calling Sequence

M.RCVR *recvaddr*

(or)

```
LA   R7,recvaddr
SVC  1,X'6B'  (or)  M.CALL H.REXS,43
```

recvaddr is the logical word address of the entry point of the receive message routine in the user's task

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the receiver address or zero if the receiver address was invalid.

6.2.96 M.READ - Read Record

The M.READ service performs the following functions:

- provides special random access handling for disk files
- unblocks system files and blocked files
- reads one record into the buffer indicated by the transfer control word (TCW) in the FCB

The base mode equivalent service is M_READ.

Entry Conditions

Calling Sequence

M.READ *fcbaddr*

(or)

```
LA   R1,fcbaddr
SVC  1,X'31'  (or)  M.CALL H.IOCS,3
```

fcbaddr is the FCB address. Appropriate transfer control parameters are defined in the TCW.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO30	ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE
IO32	CALLING TASK HAS ATTEMPTED TO PERFORM A SECOND READ ON A '$' STATEMENT THROUGH THE SYC FILE
IO33	READ WITH BYTE GRANULARITY REQUEST MADE WITH NEGATIVE BYTE OFFSET
IO34	READ WITH BYTE GRANULARITY REQUEST MADE WITHOUT SETTING RANDOM ACCESS BIT IN FCB

IO35	READ WITH BYTE GRANULARITY REQUESTS ARE VALID FOR UNBLOCKED FILES ONLY
RM02	ACCESS MODE NOT ALLOWED

Output Messages

Dismount/mount messages if EOT is encountered and if a multivolume magnetic tape.

6.2.97 M.RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode

The M.RELP service allows the privileged user to release a device from its reserved state. This service applies to dual-port extended I/O disks. When issued to an eight-line device that has been SYSGENed as full-duplex, this service can be used to set the eight-line device from single-channel to dual-channel mode (applies to ACMs using the H.F8XIO handler only).

The base mode equivalent service is M_RELP.

Entry Conditions

Calling Sequence

M.RELP *fcbaddr*

(or)

```
LA   R1,fcbaddr
SVC  1,X'27'  (or)  M.CALL H.IOCS,27
```

fcbaddr is the FCB address

Exit Conditions

Return Sequence

M.RTRN

6.2.98 M.RENAM - Rename File

The M.RENAM service changes the name of an existing permanent file. This service can move a file from one directory to another directory on the same volume.

When called, this service creates the new name of the file in the specified directory and then deletes the old name of the file from the specified directory.

The base mode equivalent service is M_RENAME.

Entry Conditions

Calling Sequence

M.RENAM [*arga*],[*newaddr*] [,*cnpaddr*]

(or)

```
LW   R1,arga
LW   R2,newaddr
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'2D'  (or) M.CALL H.VOMM,14
```

arga	contains the old PN or PNB vector
newaddr	contains the new PN or PNB vector
cnpaddr	is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA R7 (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, contains the denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.99 M.REPLAC - Replace Permanent File

The M.REPLAC service replaces the data contents of an existing permanent file with the data contents of an existing temporary file. The permanent file retains its original directory entry and resource descriptor.

This service is provided so utility programs can change the data contents of a file without changing any of the file's other attributes. In other words, this service maintains the integrity of a file's resource identifier and therefore provides the fast file mechanism.

This service can be used on any permanent file. At the completion of this service, the temporary file is deallocated and deleted. An error condition is returned if the permanent file is allocated to another task at the time of the service call.

This service should only be used on files with the fast access attribute. For files which do not have this attribute, this same functionality can be accomplished by using the Delete Resource (M.DELR) service followed by the Change Temporary File to Permanent File (M.TEMPER) service.

The base mode equivalent service is M_REPLACE.

Entry Conditions

Calling Sequence

M.REPLAC [*fcbaddr*],[*pnaddr*][,*cnpaddr*]

(or)

```
LA   R1,fcbaddr
LW   R2,pnaddr
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'30'  (or) M.CALL H.VOMM,23
```

fcbaddr	is the FCB or LFC address of the temporary file, or the value in R1, if not supplied
pnaddr	is the pathname vector of the permanent file, or the value in R2, if not supplied
cnpaddr	is a CNP address or zero if CNP is not supplied. Bit 0, word 2 of the CNP is disabled for M.REPLAC.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

CC1 set	Error condition
R7	contains the return status if a CNP is not supplied; otherwise, it is unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

Abort Cases

VM38	REPLACEMENT FILE IS ALLOCATED BY ANOTHER TASK AND BIT 0 IN THE CNP OPTION FIELD IS NOT SET, OR FILE IS ALLOCATED BY OTHER CPU IN MULTI-PORT ENVIRONMENT

6.2.100 M.RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode

The M.RESP service allows the privileged user to reserve a device to the requesting CPU until such time as a release (M.RELP) is issued. This service applies to dual-port extended I/O disks. When issued to an ACM that has been SYSGENed as full-duplex, this service can reset the ACM from dual-channel to single-channel mode (applies to ACMs using the H.F8XIO handler only).

The base mode equivalent service is M_RESP.

Entry Conditions

Calling Sequence

M.RESP *fcbaddr*

(or)

```
LA   R1,fcbaddr
SVC  1,X'26'  (or)  M.CALL  H.IOCS,24
```

fcbaddr is the FCB address

Exit Conditions

Return Sequence

M.RTRN

6.2.101 M.REWRIT - Rewrite Descriptor

The M.REWRIT service writes a modified resource descriptor back to a volume and releases the modify lock on the descriptor. This is the last step of a two step operation; the first step is M.MOD.

When this service is invoked, the indicated resource descriptor is read into an internal buffer. The fields that are allowed to be modified are copied from the user supplied resource descriptor buffer to the appropriate areas of the internal buffer. Upon successful modification of the resource descriptor in the internal buffer, the resource descriptor is written to the correct location on the volume and the modify lock is released.

The base mode equivalent service is M_REWRIT.

Entry Conditions

Calling Sequence

M.REWRIT *rdaddr*[,*cnpaddr*]

(or)

```
LA   R6,rdaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'2B'  (or) M.CALL  H.VOMM,12
```

rdaddr is the address of the RD buffer, doubleword bounded and 192W in length, containing the modified resource descriptor

cnpaddr is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, contains the denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.102 M.REWRTU - Rewrite Descriptor User Area

The M.REWRTU service writes a modified resource descriptor back to a volume and releases the modify lock on the descriptor. This is the last step of a two step operation; the first step is M.MODU.

When this service is invoked, the indicated resource descriptor is read into an internal buffer. The data from the buffer supplied by the user is then copied to the appropriate areas of the internal buffer. Upon successful modification of the resource descriptor in the internal buffer, the resource descriptor is written to the correct location on the volume and the modify lock is released.

The base mode equivalent service is M_REWRTU.

Entry Conditions

Calling Sequence

M.REWRTU [*uaaddr*][,*cnpaddr*]

(or)

```
LA   R6,uaaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'32'    (or) M.CALL  H.VOMM,27
```

uaaddr is the address of the buffer containing the modified user area. This must be the same address supplied by the caller for use with the associated Modify Descriptor User Area (H.VOMM,26) call; doubleword bounded and 32W length.

cnpaddr is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, contains the denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.103 M.ROPL - Reset Option Lower

The M.ROPL service allows the calling task to reset the option lower bit. Use the M.SOPL (Set Option Lower) service to set the option lower bit.

The base mode equivalent service is M_ROPL.

Entry Conditions

Calling Sequence

M.ROPL

(or)

SVC 2,X'78' (or) M.CALL H.TSM,14

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN (CC1 set)

Status

CC1 set call caused the option lower bit to be reset

6.2.104 M.RRES - Release Channel Reservation

The M.RRES service releases the channel reserve indication from the controller definition table (CDT) entry if the channel was reserved by the calling task. If the channel was not reserved by the calling task, the request is ignored and control returns to the task.

The base mode equivalent service is M_RRES.

Entry Conditions

Calling Sequence

M.RRES *channel*

(or)

```
LW   R1,channel
SVC  1,X'3B'  (or)  M.CALL  H.IOCS,13
```

channel specifies the channel number in hexadecimal. If using LW, the channel number must be loaded into bits 24 to 31 of R1.

Exit Conditions

Return Sequence

M.RTRN

6.2.105 M.RSML - Resourcemark Lock

The M.RSML service locks the specified resourcemark. It is used with the Unlock Resourcemark service (M.RSMU) by tasks to synchronize access to a common resource.

The base mode equivalent service is M_RSML.

Entry Conditions

Calling Sequence

M.RSML *lockid*,[*timev*][,P]

(or)

```
 LI     R4,timev
 ZR     R5
[ SBR   R5,0 ]
 LI     R6,lockid
 SVC    1,X'19'  (or)  M.CALL H.REXS,62
```

lockid is a numeric resourcemark index value, 33 through 64 inclusive

timev is a numeric value which specifies the action to be taken if the lock is already set and is owned by another task:

Value	Description
+1	immediate denial return
0	wait until this task is the lock owner (default)
-*n*	wait until this task is the lock owner, or until *n* timer units have expired, whichever occurs first

If not specified, zero is default.

P specifies that while this task is waiting to become lock owner, the swapping mode is to be set to swap this task only if a higher priority task is requesting memory space.

Otherwise, the task is a swap candidate if any task is requesting memory.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains zero if the request was accepted, otherwise contains a request denial code:

Value	Description
1	lock index exceeds maximum range
2	lock index is less than minimum range
3	lock is owned by another task (and *timev*=+1)
4	lock is owned by another task, *timev*=-*n* and *n* timer units have expired

6.2.106 M.RSMU - Resourcemark Unlock

The M.RSMU service unlocks a resourcemark which was locked by a call to the M.RSML service. If any other tasks are waiting to lock the specified resourcemark, the highest priority waiting task becomes the new lock owner.

The base mode equivalent service is M_RSMU.

Entry Conditions

Calling Sequence

M.RSMU *lockid*

(or)

```
LI    R6,lockid
SVC   1,X'1A'  (or)  M.CALL  H.REXS,63
```

lockid is a numeric resourcemark index value, 33 through 64 inclusive

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains zero if the request was accepted, otherwise contains a request denial code:

Value	Description
1	lock index exceeds maximum range
2	lock index is less than minimum range
3	lock is not owned by this task

6.2.107 M.RSRV - Reserve Channel

The M.RSRV service reserves a channel by inserting the task number of the calling task in the controller definition table (CDT) of the unit definition table (UDT). The calling task must be privileged. If the task is unprivileged or the channel is already reserved by another task, this service makes a denial return. If any requests are currently queued for this channel, suspend is invoked until any I/O currently in progress is complete. The standard handler is then disconnected from the service interrupt (SI) level. After reserving a channel, the task must connect its own handler to the SI dedicated location.

This service is not applicable for extended I/O channels.

The base mode equivalent service is M_RSRV.

Entry Conditions

Calling Sequence

M.RSRV *channel,denial*

(or)

```
LW   R1,channel
LA   R7,denial
SVC  1,X'3A'  (or)  M.CALL  H.IOCS,12
```

channel is the hexadecimal channel number in bits 24 to 32. If using LW, load channel number in R1.

denial is the user's denial return address

Exit Conditions

Return Sequence

M.RTRN normal return

(or)

M.RTNA 7 denial return

6.2.108 M.RWND - Rewind File

The M.RWND service performs the following functions:

- issues an end-of-file and purge if the file is a system or blocked file that is output active
- for system and blocked files, initializes blocking buffer control cells for subsequent access
- rewinds a file or device

The base mode equivalent service is M_REWIND.

Entry Conditions

Calling Sequence

M.RWND *fcb*

(or)

```
LA   R1,fcb
SVC  1,X'37'  (or)  M.CALL  H.IOCS,2
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09     ILLEGAL OPERATION ON THE SYC FILE
```

6.2.109 M.SETS - Set User Status Word

The M.SETS service allows the calling task to modify any task's user status word. It allows task-to-task communication when used with the M.TSTS (Test User Status Word) service. The user status word is in the CPU dispatch queue (DQE.USW) and has a value of zero until modified by this service. The service removes the user status word from the queue, modifies it as specified, and replaces it in the queue.

The base mode equivalent service is M_SETS.

Entry Conditions

Calling Sequence

M.SETS *function,statusw*[,*task*]

(or)

```
LD   R4,task
LI   R6,function
LW   R7,statusw
SVC  1,X'48'  (or)  M.CALL H.REXS,7
```

function is the type of modification to perform. Valid values are:

Value	Description
STF(1)	set flag
RSF(2)	reset flag
STC(3)	set counter
INC(4)	increment counter

If using the macro call, the alphabetical code must be specified. If loading registers, the corresponding numeric must be specified.

statusw contains a function parameter specific to function codes as follows:

Value	Description
1	bit position in the status word to be set (1-31)
2	bit position in the status word to be reset (1-31)
3	value to set the status word
4	value to increment the status word

task is the address of a doubleword containing the 1 to 8 ASCII character name of the task, left justified and blank filled, or zero in word 0 and the task number in word 1. The task number must be used if the task is multicopied or shared. A task number of zero or omission of the argument specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 5

Registers

R5 bit 0 is set if the specified task was not found in the dispatch queue or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by M.KEY file; otherwise, R5 is zero

Abort Cases

RX05 INVALID FUNCTION CODE HAS BEEN SPECIFIED FOR REQUEST TO SET USER STATUS WORD

6.2.110 M.SETSYNC - Set Synchronous Resource Lock

The M.SETSYNC service is used with the Release Synchronous Lock service for resource gating of explicitly shared resources when no automatic synchronization is performed by the system. The mechanism allows a task to obtain synchronized access to a resource that is concurrently allocated to multiple tasks. A synchronization lock can be obtained for any resource, provided the resource was previously allocated, or included for memory partitions by the calling task. Unlike an exclusive lock, the synchronous lock does not prevent other tasks from allocating the resource in explicit shared mode. It is the sharing tasks' responsibility to synchronize access by cooperative use of the synchronous lock services. The resource is identified by either a logical file code, defined when the resource was assigned, or an allocation index, obtained when the resource was assigned or by a resource inquiry. If the synchronization lock is not available, the calling task can obtain an immediate denial return, or wait for an indefinite or specified period of time.

The base mode equivalent service is M_SETSYNC.

Entry Conditions

Calling Sequence

M.SETSYNC *arga*[,*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'46'   (or) M.CALL  H.REMM,25
```

arga is an address the allocation index obtained when the resource was assigned

(or)

is an address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC was not assigned by this task
30	invalid allocation index
38	time out occurred while waiting to become lock owner
46	unable to obtain resource descriptor lock (multiprocessor only)
50	resource is locked by another task

Wait Conditions

The task is placed in a wait state, as appropriate, if specified in the CNP.

6.2.111 M.SETT - Create Timer Entry

The M.SETT service builds an entry in the timer table so that the requested function is performed upon time out. Timer entries can be created to activate a program, resume a program, set a bit in memory, reset a bit in memory, or request an interrupt. Any task can create a timer to activate or resume a program. Any task can create a timer entry to set or reset bits provided the bit is within a static memory partition. Only privileged tasks can set bits in the operating system and request an interrupt.

The base mode equivalent service is M_SETT.

Entry Conditions

Calling Sequence

M.SETT *timer,t1,t2,function,arg4,arg5*

(or)

```
LB          R3,function
SLL         R3,24
ORMW        R3,timer
LW          R4,t1
LW          R5,t2
LW (or LD)  R6,arg4
(LW         R7,arg5)
SVC         1,X'45'  (or)  M.CALL H.REXS,4
```

timer is a word containing zeros in bytes 0 and 1, and a 2-character timer identification in bytes 2 and 3

t1 is the current value the timer will be set to in negative time units

t2 is the value the timer will be reset to upon each time out in negative time units. If the reset value is zero, the function is performed upon time out and the timer entry is deleted. This case is called a one-shot timer entry.

function is the function to be timed, as follows:

Function	Description
ACP(1)	activate program
RSP or RST(2)	resume program
STB(3)	set bit
RSB(4)	reset bit
RQI(5)	request interrupt

If using the macro call, the alphabetic code must be specified. If loading registers, the corresponding numeric must be specified in byte 0.

The function code and *arg4* and *arg5* contain values specific to the function being timed as follows:

Function Code		
Alphabetic	Numeric	*arg4* and *arg5*
ACP	1	*arg4* is a doubleword containing the 1- to 8-character name of the program to be activated if system file, or pathname vector or RID vector in the first half of the doubleword (R6) and zero in the second half of the doubleword (R7). If the task named is not currently in execution, it is preactivated to connect the interrupt to the task. This connection remains in effect until the task aborts or the timer is deleted. On normal exit, the timer table is updated to point to the next generation. *arg5* is null.
RSP	2	*arg4* is a doubleword containing the 1- to 8-character name of the task to be resumed or the task number in R7 and zero in R6. *arg5* is null.
(or)		
RST	2	*arg4* is the task number in R7 and zero in R6. *arg5* is null.
STB	3	*arg4* is the address of the word in which the bits are to be set. The address must be in a static memory partition or the operating system. *arg5* is the bit configuration of the mask word to be ORed.
RSB	4	*arg4* is the address of the word in which the bit is to be reset. The address must be in a static memory partition or the operating system. *arg5* is the bit configuration of the mask word to be ANDed.
RQI	5	*arg4* is the priority level of the interrupt to be requested. *arg5* is null.

Exit Conditions

Return Sequence

M.RTRN R3

R3 unchanged and condition codes are not set

Error Condition

M.RTRN R3

If there are no timer entries available, R3 is zero and condition codes are not set.

If there are timer entries available, R3 is zero and one of the following condition codes are set:

- CC1 set if requested load module does not exist, or the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name
- CC2 set if requested task is not active
- CC3 set if attempting to create a duplicate timer ID

Abort Cases

RX02 INVALID FUNCTION CODE SPECIFIED FOR REQUEST TO CREATE A TIMER ENTRY. VALID CODES ARE ACP(1), RSP OR RST(2), STB(3), RSB(4) AND RQI(5).

RX03 TASK ATTEMPTED TO SET/RESET A BIT OUTSIDE OF A STATIC PARTITION OR THE OPERATING SYSTEM

RX04 THE REQUESTING TASK IS UNPRIVILEGED OR HAS ATTEMPTED TO CREATE A TIMER ENTRY TO REQUEST AN INTERRRUPT WITH A PRIORITY LEVEL OUTSIDE THE RANGE OF X'12' TO X'7F', INCLUSIVE

6.2.112 M.SMSGR - Send Message to Specified Task

The M.SMSGR service allows a task to send up to 768 bytes to the specified destination task. Up to 768 bytes can be accepted as return parameters.

The base mode equivalent service is M_SMSGR.

Entry Conditions

Calling Sequence

M.SMSGR *psbaddr*

(or)

```
LA   R2,psbaddr
SVC  1,X'6C'  (or)  M.CALL  H.REXS,44
```

psbaddr is the logical address of the parameter send block (PSB).

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing initial error status if any:

Value	Description
0	normal initial status
1	task not found or the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name
2-9	reserved
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task queue depth exceeded
16	invalid PSB address

6.2.113 M.SOPL - Set Option Lower

The M.SOPL service allows the calling task to set the option lower bit. Use the M.ROPL (Reset Option Lower) service to reset the option lower bit.

The base mode equivalent service is M_SOPL.

Entry Conditions

Calling Sequence

M.SOPL

(or)

SVC 2,X'77' (or) M.CALL H.TSM,13

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN (CC1 set)

Registers

CC1 set call caused the option lower bit to be set

6.2.114 M.SRUNR - Send Run Request to Specified Task

The M.SRUNR service allows a task to activate or reexecute the specified destination task with a parameter pass of up to 768 bytes. Up to 768 bytes can be accepted as return parameters.

When a task name is supplied in words 0 and 1 of the parameter send block (PSB), the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or an RID vector must be supplied.

If a task activated with the TSM ACTIVATE directive is sent a run request, the queued run request is ignored. However, if a task is activated with a run request and a second run request is sent, the queued run request is then executed.

The base mode equivalent service is M_SRUNR.

Entry Conditions

Calling Sequence

M.SRUNR *psbaddr*

(or)

```
LA   R2,psbaddr
SVC  1,X'6D'  (or)  M.CALL  H.REXS,45
```

psbaddr is the logical address of the parameter send block (PSB).

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 contains the processing initial error status if any:

Value	Description
0	normal initial status
1	reserved
2	file specified in the PSB was not found in directory
3	reserved
4	file specified in the PSB is not a load module or executable image
5	dispatch queue entry (DQE) unavailable
6	I/O error on directory read
7	I/O error on load module read
8	memory unavailable

Value	Description
9	invalid task number for run request to multicopied load module in RUNW state
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task queue depth exceeded
16	invalid PSB address
17	reserved

R7 contains the task number of the destination task, or zero if the request was not processed

6.2.115 M.SUAR - Set User Abort Receiver Address

The M.SUAR service specifies an address where control is to return if an abort condition occurs during task execution.

All files remain open prior to transferring to the specified address.

The base mode equivalent service is M_SUAR.

Entry Conditions

Calling Sequence

M.SUAR *address*

(or)

```
LA   R7,address
SVC  1,X'60'  (or)  M.CALL H.REXS,26
```

address is the logical address where control is transferred when a task terminates

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 bit 0 is zero if the request is honored, or one if the request is denied because the specified address is outside the user's allocated area; bits 1-31 are unchanged

Abort Cases

```
RX89    AN UNPRIVILEGED TASK HAS ATTEMPTED TO REESTABLISH
        AN ABORT RECEIVER (OTHER THAN M.IOEX)
```

6.2.116 M.SUME - Resume Task Execution

The M.SUME service resumes a task that has been suspended. A request to resume a task which is not suspended is ignored.

The base mode equivalent service is M_SUME.

Entry Conditions

Calling Sequence

M.SUME *task*

(or)

```
ZR   R6          } (or) LD  R6,taskname
LW   R7,taskno   }
SVC  1,X'53'   (or) M.CALL  H.REXS,16
```

task the address of a doubleword containing the name of a task or zero in word 0 and the task number in word 1. Task number must be used if the task is multicopied or shared.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted by M.KEY file from access to tasks with a different owner name; otherwise, unchanged

6.2.117 M.SURE - Suspend/Resume

The M.SURE service suspends the calling task and resumes the target task. The suspend and resume functions are combined into one module providing faster context switching. It does not replace M.SUSP and M.SUME.

M.SURE applies to real-time and time distribution tasks with the target task priority higher than or equal to that of the calling task. Context switch time can be further improved by turning real-time accounting off. This service is not recommended for two IPU biased tasks.

The accounting option is turned off using the OFRA option to the MODE directive in SYSGEN or OPCOM and the CATALOG ENVIRONMENT directive.

The base mode equivalent service is M_SURE.

Entry Conditions

Calling Sequence

M.SURE *taskno*

(or)

```
LW   R7,taskno
SVC  5,X'00'
```

taskno is the task number of the target task

Exit Conditions

Return Sequence

No return. All registers are destroyed. When the service completes normally, CC1 is reset. The next instruction is in the target task.

Abnormal Return:

CC1 set

R7 contains a code describing the reason for the error:

Value	Description
1	task not found
2	task not in suspend state
3	owner/access violation

Return is done via LPSD to the calling task.

Attempts to execute this service when the module H.SURE is not configured will abort the calling task with an SV02 abort code.

6.2.118 M.SUSP - Suspend Task Execution

The M.SUSP service suspends the calling task or any other specified task for the specified number of time units or for an indefinite time period. Suspending a task for a time interval results in a one-shot timer entry to resume the task upon time-out of the specified interval. A task suspended for an indefinite time interval must be resumed through the M.SUME system service. A suspended task can also be resumed upon receipt of a message interrupt. A message sent to a task that is synchronized (M.SYNCH) and suspended is not received, but the task is resumed.

The base mode equivalent service is M_SUSP.

Entry Conditions

Calling Sequence

M.SUSP *task,time1*

(or)

```
LW   R5,time1
LI   R6,0          } (or) LD R6,taskname
LW   R7,taskno     }
SVC  1,X'54'  (or) M.CALL  H.REXS,17
```

task is the address of a doubleword containing the name of a task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

time1 is zero, if requesting suspension for an indefinite time interval, or the negative number of time units to elapse before the calling task is resumed

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name; otherwise, contains the task number

(or)

contains zero and CC1 is set if the specified taskname is multicopied

6.2.119 M.SYNCH - Set Synchronous Task Interrupt

The M.SYNCH service causes message and task interrupts to be deferred until the user makes a call to M.ANYW, M.EAWAIT, M.WAIT, or M.ASYNCH. When this service is used, message interrupts are not interrupted by end-action interrupts. All task interrupt levels cannot be interrupted, except by break, until they voluntarily relinquish control.

If a synchronized task is suspended then a message is sent to the task, the message receiver is not entered and the task resumes.

This service can be executed by the IPU.

The base mode equivalent service is M_SYNCH.

Entry Conditions

Calling Sequence

M.SYNCH

(or)

SVC 1,X'1B' (or) M.CALL H.REXS,67

Exit Conditions

Return Sequence

M.IPURTN

Registers

CC1 set synchronous task interrupt was already set

6.2.120 M.TBRKON - Trap Online User's Task

The M.TBRKON service processes a pause or break from the terminal or calling task. The service is also the default receiver for any online task and is called as a result of a hardware or software break. If a transfer control word (TCW) is specified, a user message is printed with the break message. Refer to the description of FCB word 1 in Chapter 5 for more information on TCWs.

The base mode equivalent service is M_TBRKON.

Entry Conditions

Calling Sequence

M.TBRKON *tcw*

(or)

```
LW   R2,tcw (or)   ZR  R2
SVC  1,X'5C' (or)  M.CALL  H.TSM,6
```

tcw is the address of a transfer control word

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
RX34    TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO
        BREAK IS ACTIVE
TS01    USER REQUESTED REMOVAL FROM A BREAK REQUEST
```

6.2.121 M.TDAY - Time-of-Day Inquiry

The M.TDAY service obtains the time-of-day as computed from the real-time clock interrupt counter. The counter is initialized using a SYSGEN parameter.

This service can be executed by the IPU.

The base mode equivalent service is M_TDAY.

Entry Conditions

Calling Sequence

M.TDAY

(or)

SVC 1,X'4E' (or) M.CALL H.REXS,11

Exit Conditions

Return Sequence

M.IPURTN 7

Registers

R7 Byte	Contents
0	hours (0 to 23)
1	minutes (0 to 59)
2	seconds (0 to 59)
3	interrupts (less than one second)

6.2.122 M.TEMP - Create Temporary File

The M.TEMP service creates a temporary file. Temporary files are not given names in directories and remain known to the operating system only for as long as the task that created them is in execution. Typically, when the task that created a temporary file terminates execution either normally or abnormally, associated temporary files are automatically deleted by the operating system.

Temporary files can remain defined to the operating system after the task that created them terminates execution if the temporary file is made permanent or is allocated or assigned to another task when the creator terminates execution.

This service allocates a resource descriptor for the file and acquires the initial space requirements for the file. The attributes of the file are then recorded in the resource descriptor.

When a temporary file is created, the typical file attributes are:

- resource identifier (RID)
- protection attributes
- management attributes
- initial space requirements

The file's RID is returned only if an RCB address is specified and an ID address for the file is also specified within the RCB.

To create a file with possible multiple segments, the CNP address must be supplied. Byte 0 of the CNP option field contains the maximum number of segments allowed at creation. If M.TEMP creates a file with one segment, and the size of the created file is less than the size requested, condition code bit 1 is set and status is returned in the CNP.

Asynchronous abort and delete are inhibited during execution of this service.

The base mode equivalent service is M_CREATET.

Entry Conditions

Calling Sequence

M.TEMP [*cnpaddr*],[*arga*][,*rcbaddr*]

(or)

```
LW   R1,arga      (or) ZR  R1
LA   R2,rcbaddr   (or) ZR  R2
LA   R7,cnpaddr   (or) ZR  R7
SVC  2,X'21'      (or) M.CALL  H.VOMM,2
```

cnpaddr	is a CNP address or zero if CNP not supplied
arga	contains a PN (volume name only) vector. If *arga* is not specified, the file is created on the current working volume.
rcbaddr	is a RCB address or not specified if default attributes are desired

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.123 M.TEMPER - Change Temporary File to Permanent File

The M.TEMPER service makes a temporary file permanent. The temporary file is given a name in the specified directory and the file's resource type is changed from temporary to permanent. The file is made permanent with the attributes that were defined when it was created and with any new attributes that were acquired while the file's data was being established, such as additional extensions, end-of-file position, or explicit resource descriptor modifications incurred prior to invocation of this service. The temporary file can be made permanent only on the volume where the temporary file resides, i.e., cross volume definitions are not allowed.

This service ensures exclusive use of a file while the initial file data is being established. The integrity of the file is guaranteed before the file is defined in a directory where others can gain access to it.

When the directory entry is established, it is linked to the resource descriptor of the file. This link relates the name of the file to the other attributes of the file. These attributes are the same as the attributes for a permanent file.

The base mode equivalent service is M_TEMPFILETOPERM.

Entry Conditions

Calling Sequence

M.TEMPER [*arga*],[*argb*][,*cnpaddr*]

(or)

```
LW   R1,arga
LW   R2,argb
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'28'    (or) M.CALL  H.VOMM,9
```

arga	is an LFC or an FCB address
argb	is a PN vector or PNB vector
cnpaddr	is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.124 M.TRNC - Truncate File

The M.TRNC service truncates the unused space of a file. This service is the complement of the extend service (M.EXTD). Only manually extended files need to be truncated.

This service truncates only temporary or permanent files. Directories and memory partitions cannot be truncated. The caller must have write, update or append access to truncate the file. A file cannot be truncated to less than the minimum space requirement of the file as defined when the file was created.

A file can be truncated regardless of whether it is currently allocated. Any allowable resource specification can be supplied for example, pathname (PN), pathname block (PNB), resource ID (RID), logical file code (LFC), or address of a file control block (FCB).

Asynchronous abort and delete are inhibited during execution of this service.

The base mode equivalent service is M_TRUNCATE.

Entry Conditions

Calling Sequence

M.TRNC [*arga*][,*cnpaddr*]

(or)

```
LW   R1,arga
LA   R7,cnpaddr    (or) ZR  R7
SVC  2,X'26'       (or) M.CALL  H.VOMM,7
```

arga	contains a PN vector, a PNB vector, an RID vector, an LFC, or an FCB
cnpaddr	is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.2.125 M.TSCAN - Scan Terminal Input Buffer

The M.TSCAN service parses the line buffer pointed to by T.LINBUF. The service is used by tasks to scan a line of terminal input. The parameters (fields) to be scanned are in the user's line buffer. Each call to M.TSCAN returns one parameter from the line buffer and updates the current scan position. When a call returns a zero in R5 and a carriage return in R4, end of line (EOL) has been reached. Each read from the terminal reinitializes the line buffer and the pointer.

The base mode equivalent service is M_TSCAN.

Entry Conditions

Calling Sequence

M.TSCAN

(or)

SVC 1,X'5B' (or) M.CALL H.TSM,2

Exit Conditions

Return Sequence

M.RTRN 4,5,6,7

(or)

M.RTRN CC1 set if a line buffer is not found.

Registers

R4	contains the delimiting character; carriage return if CC1 set
R5	number of significant characters before delimiter; zero if CC1 set
R6,7	first eight characters of the character string, left-justified. The entire character string is in words 0 through 3 of the terminal line buffer. For nonbase mode tasks, the terminal line buffer address can be obtained by accessing the T.LINBUF variable in the TSA.

Notes:

M.TSCAN ignores all blanks encountered before the first parameter or delimiter. If M.TSCAN encounters a delimiter before the first parameter, it continues to ignore all blanks until encountering the first parameter.

The M.RWND service resets the cursor at the first parameter of the current input line. M.TSCAN scans the line without any additional IOCS calls.

6.2.126 M.TSMPC - TSM Procedure Call

The M.TSMPC service receives TSM procedure call directive strings and returns the results of the directive or an error message to the user-supplied buffer. The service supports the following procedure call directives: $BATCH, $DIRECTORY, $ERR, $GETPARM, $LINESIZE, $PAGESIZE, $PROJECT, $RRS, $SET, $SETI, $TABS, and $VOLUME.

The maximum size input string is 72 characters. The size of the output string depends on the input directive as follows:

Directive	Maximum Output
$BATCH	no output for normal processing
$DIRECTORY	16-character directory name from the M.KEY file
$ERR	212 characters and two carriage control characters (CR/LF) per line for an abort code definition. The ASCII control characters for LF and CR delimit the lines returned from $ERR.
$GETPARM	72 characters for a parameter value. If the specified parameter exists but has not received a value, the parameter name is returned. If the parameter does not exist, an error message is returned.
$LINESIZE	no output for normal processing
$PAGESIZE	no output for normal processing
$PROJECT	8-character project name from the M.KEY file
$RRS	variable length RRS entry for the user-supplied LFC assignment
$SET/$SETI	no output for normal processing
$TABS	8 tab settings from the M.KEY file
$VOLUME	16-character volume name from the M.KEY file

Error messages are a maximum of 80 characters and two ASCII control characters (CR/LF) as EOL delimiters.

R7 must be zero on entry to this service.

Refer to the Notes section below for information on the syntax of the directives.

The base mode equivalent service is M_TSMPC.

Entry Conditions

Calling Sequence

M.TSMPC *pcb*

(or)

```
LA   R1,pcb
ZR   R7
SVC  2,X'AE'   (or)   M.CALL H.TSM,17
```

pcb is the address of a 4-word procedure call block (PCB)

Procedure Call Block (PCB)

The PCB contains the information necessary for the service to complete a procedure call. The format of the PCB is as follows:

	0 7	8 15	16 23	24 31
Word 0		Send buffer address (PCB.SBA)		
1	Send quantity (PCB.SQUA)			
2		Return buffer address (PCB.RBA)		
3	Actual return length (PCB.ACRP)		Return buffer length (PCB.RPBL)	

Send buffer address	is the address of a character string that represents a valid TSM procedure call directive
Send quantity	contains the length in bytes of the TSM procedure call directive
Return buffer address	is the address of a buffer to contain either valid return information or an error message if CC1 is set and R7 contains a value of 1
Actual return length	is the number of bytes returned from the procedure call
Return buffer length	is the size of the supplied return buffer

Exit Conditions

Return Sequence

M.RTRN

or

M.RTRN R7 (CC1 set)

Registers

R7 Return status if error; otherwise, zeroed.

Status

CC1 set

Posted in R7

Value	Description
1	return buffer contains error message
2	invalid send buffer address
3	send buffer size is zero
4	send buffer too long
5	invalid return buffer address
6	return buffer size is zero
7	return data has been truncated
8	invalid PCB address
9	invalid SVC from a non-TSM task

Note: For the syntax of the $BATCH, $ERR, $LINESIZE, $PAGESIZE, $SET, and $SETI directives, refer to the MPX-32 Reference Manual Volume II, Chapter 1. The $RRS directive is similar to the $ASSIGN directive. Refer to the $ASSIGN directive syntax in the MPX-32 Reference Volume II, Chapter 1 and specify $RRS rather than $ASSIGN. The syntax for $DIRECTORY, $PROJECT, $TABS, and $VOLUME is the directive name or its 4-character abbreviation plus $ if used. These directives display information only. The syntax of the $GETPARM directive is as follows:

$GETPARM *parm*

parm is the name of a parameter defined in the directive file associated with the task. No percent sign precedes the parameter name.

6.2.127 M.TSTE - Arithmetic Exception Inquiry

The M.TSTE service resets the arithmetic exception status bit in the user's TSA and returns CC1 set or reset according to the status value. The status bit is set whenever the user is in execution and an arithmetic exception trap occurs. The bit remains set until this service is requested or the task terminates.

This service can be executed by the IPU.

The base mode equivalent service is M_TSTE.

Entry Conditions

Calling Sequence

M.TSTE

(or)

SVC 1,X'4D' (or) M.CALL H.REXS,23

Exit Conditions

Return Sequence

M.IPURTN

None

Registers

PSD CC1 contains the value of the arithmetic exception status bit

6.2.128 M.TSTS - Test User Status Word

The M.TSTS service returns the 32-bit user status word of any executing task. The user status word resides in the CPU dispatch queue (DQE.USW) and is modified by the Set User Status Word (M.SETS) system service. These services treat the user status word as either a set of 32 flags or as a 32-bit counter. Bit 0 is used as a status flag.

The base mode equivalent service is M_TSTS.

Entry Conditions

Calling Sequence

M.TSTS *task*

(or)

```
ZR   R6          } (or) LD R6,taskname
LW   R7,taskno   }
SVC  1,X'49'   (or)  M.CALL  H.REXS,8
```

task the address of a doubleword containing the name of a task or zero in word 1 and the task number in word 2. Task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 bit 0 is set if the specified task was not found or the requesting task is not privileged and the owner name is restricted by the M.KEY file from access to tasks with a different owner name; otherwise, R7 returns the user-status word

6.2.129 M.TSTT - Test Timer Entry

The M.TSTT service returns to the caller the negative number of time units remaining until the specified timer entry timeout. If the timer has expired, zero is returned.

This service can be executed by the IPU.

The base mode equivalent service is M_TSTT.

Entry Conditions

Calling Sequence

M.TSTT *timer*

(or)

```
LW   R6,timer
SVC  1,X'46'  (or)  M.CALL   H.REXS,5
```

timer is the 2-character ASCII name of a timer, right-justified

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the negative number of time units remaining until time out or contains zero if the timer has expired or does not exist

6.2.130 M.TURNON - Activate Program at Given Time-of-Day

The M.TURNON service activates or resumes a specified task at a specified time and reactivates or resumes it at specified intervals. M.TURNON creates a timer table entry using a specified timer ID. When a load module or executable image name is supplied, MPX-32 defaults to a search in the system directory only. For activations in other than the system directory, a pathname or an RID vector must be supplied.

The base mode equivalent service is M_TURNON.

Entry Conditions

Calling Sequence

M.TURNON *filename,time,[reset],timerid*

(or)

```
LD   R6,filename
LW   R4,time
LW   R5,reset
LW   R3,timerid
SVC  1,X'1E'   (or)   M.CALL   H.REXS,66
```

filename is a left-justified blank-filled doubleword containing the 1- to 8-character ASCII name of a system load module, executable image file, the pathname vector, or RID vector pointing to the task to be activated in R6, and zero in R7.

time is the time-of-day on the 24-hour clock when the task is activated. It is a word value with the following format:

Byte	Contents in Binary
0	hours
1	minutes
2	seconds
3	zero

reset is the time interval on the 24-hour clock to elapse before resetting the clock upon each time out. It has the same format as the time argument above. The task is reactivated at each time out. If a reset value is not specified, the comma denoting the field must still be specified and the task is activated only once.

timerid is a word variable containing the right-justified, zero-filled, 2-character ASCII name of the timer that will be created

Exit Conditions

Return Sequence

M.RTRN R3 nonzero

Error Condition

M.RTRN	R3	zero if there are no timer entries available, the requested load module or executable image does not exist, requested duplicate timer IO already exits or invalid timer ID

6.2.131 M.TYPE - System Console Type

The M.TYPE service types a user specified message and optionally reads from the system console. Input message address is validated for the unprivileged task. The operation is wait I/O.

If a response is not detected within 30 seconds of an M.TYPE, the M.TYPE terminates with 0 bytes transferred if there is any queued console I/O. If a response is detected within 30 seconds of an M.TYPE, the read does not timeout.

The maximum input or output is 80 characters. If no characters are specified, the maximum is used.

M.TYPE builds a type control parameter block (TCPB) that defines the I/O buffer 24-bit addresses for console messages and reads.

The base mode equivalent service is M_TYPE.

Entry Conditions

Calling Sequence

M.TYPE *outmess,outcount*[,*inmess,incount*]

(or)

```
LA   R1,tcpb
SVC  1,X'3F' (or)  M.CALL  H.IOCS,14
```

outmess	is the 24-bit address of the output message buffer
outcount	is the transfer count for output. Up to 80 bytes can be transferred.
inmess	is the 24-bit address of the input message buffer. If not specified, TCPB word 2 is zeroed. The first byte of this field contains the actual input quantity.
incount	is the transfer count for input. Up to 80 bytes can be transferred.
tcpb	is the address of the TCPB

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO15	A TASK HAS REQUESTED A TYPE OPERATION AND THE TYPE CONTROL PARAMETER BLOCK (TCPB) SPECIFIED INDICATES THAT AN OPERATION ASSOCIATED WITH THAT TCPB IS ALREADY IN PROGRESS

6.2.132 M.UNLOCK - Release Exclusive Resource Lock

The M.UNLOCK service releases an exclusive resource lock set with the Set Exclusive Resource Lock (M.LOCK) service. An exclusive resource lock can not be released by a task other than the owning task. When called, the exclusive lock is released if the task allocated the resource in a shareable mode; otherwise, the lock cannot be released until the resource is deallocated. Once the lock is released, other tasks can allocate the resource in a compatible access mode for the particular shared usage. However, another task cannot exclusively lock the resource until this task, and all other sharing tasks, deallocate the resource. Any outstanding exclusive resource locks are released on task termination or on resource deallocation.

The base mode equivalent service is M_UNLOCK.

Entry Conditions

Calling Sequence

M.UNLOCK *arga*[,*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr     (or) ZR  R7
SVC  2,X'45'        (or) M.CALL  H.REMM,24
```

arga is the address of the allocation index obtained when the resource was assigned (or) the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC not assigned by this task
30	invalid allocation index
32	an exclusive resource lock was not owned by this task
33	resource is not allocated in a shareable mode by this task
46	unable to obtain resource descriptor lock (applicable for a multiprocessor only)

6.2.133 M.UNSYNC - Release Synchronous Resource Lock

The M.UNSYNC service is used with the Set Synchronous Resource Lock (M.SETSYNC) service to perform gating on resources allocated for explicit shared use. When the service is called, the synchronization lock is released, and all tasks waiting to own the lock are polled.

A synchronization lock may not be cleared by any task other than the one that set the lock.

A synchronization lock is automatically released when a task terminates or deallocates the resource.

The base mode equivalent service is M_UNSYNC.

Entry Conditions

Calling Sequence

M.UNSYNC *arga*[,*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr (or) ZR R7
SVC  2,X'47'  (or)  M.CALL H.REMM,26
```

arga is the address of the allocation index obtained when the resource was assigned

(or)

is the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC was not assigned by this task
30	invalid allocation index
32	synchronization lock was not set
46	unable to obtain resource descriptor lock (multiprocessor only)

6.2.134 M.UPRIV - Change Task to Unprivileged Mode

The M.UPRIV service allows a task that was cataloged as privileged to operate in an unprivileged state. This causes the calling task's protection image to be loaded at every context switch. See the M.PRIV service to reinstate privilege status.

The base mode equivalent service is M_UNPRIVMODE.

Entry Conditions

Calling Sequence

M.UPRIV

(or)

SVC 2,X'58' (or) M.CALL H.REXS,79

Exit Conditions

Return Sequence

M.RTRN

6.2.135 M.UPSP - Upspace

If the M.UPSP service writes a volume record header if bottom of tape (BOT) is encountered or performs an Erase/Write EOF if end of tape (EOT) is encountered on multivolume magnetic tape. The M.UPSP service is not applicable to blocked or SYC files.

The base mode equivalent service is M_UPSP.

Entry Conditions

Calling Sequence

M.UPSP *fcb*

(or)

```
LA   R1,fcb
SVC  1,X'10'  (or)  M.CALL  H.IOCS,20
```

fcb is the FCB address

Registers

R1 FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO06 INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.

IO09 ILLEGAL OPERATION ON THE SYC FILE

Output Messages

Mount/dismount messages if EOT on multivolume magnetic tape.

6.2.136 M.VADDR - Validate Address Range

The M.VADDR service verifies the specified logical address provided.

The base mode equivalent service is M_VADDR.

Entry Conditions

Calling Sequence

M.VADDR *addr,bytes*

(or)

```
LW   R6,addr
LI   R7,bytes
SVC  2,X'59'   (or)   M.CALL  H.REXS,33
```

addr	is the logical starting address of validation
bytes	is the number of bytes to validate

Exit Conditions

Return Sequence

M.RTRN

Registers

CC2 set	address range crosses map block boundary
CC3 set	locations specified are protected
CC4 set	invalid address (not in caller's address space)
R0-R7	unchanged

6.2.137 M.WAIT - Wait I/O

The M.WAIT service returns to the user when the I/O request associated with the specified FCB is complete. The task is suspended until I/O completes.

The base mode equivalent service is M_WAIT.

Entry Conditions

Calling Sequence

M.WAIT *fcb*

(or)

```
LA   R1,fcb
SVC  1,X'3C'  (or)  M.CALL  H.IOCS,25
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
MS31    USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN
        END-ACTION ROUTINE
```

6.2.138 M.WEOF - Write EOF

The M.WEOF service performs the following functions:

- prevents a write to a read-only file
- issues an end-of-file and purge if the file is a blocked file with an active blocking buffer
- writes a software EOF record (a 192-word record with X'0FE0FE0F' in its first word) immediately following the last record of an unblocked file created with EOFM=F
- writes volume record if BOT is encountered on multivolume magnetic tape
- performs an erase and write EOF if EOT is encountered on multivolume magnetic tape

The base mode equivalent service is M_WRITEEOF.

Entry Conditions

Calling Sequence

M.WEOF *fcb*

(or)

```
LA   R1,fcb
SVC  1,X'38'  (or)  M.CALL  H.IOCS,5
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09     ILLEGAL OPERATION ON THE SYC FILE
IO30     ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID
         ENCOUNTERED ON MAGNETIC TAPE
```

Output Messages

Dismount/mount messages if EOT encountered on multivolume magnetic tape.

6.2.139 M.WRIT - Write Record

The M.WRIT service performs the following functions:

- prevents a write to a read-only file
- provides special random access handling for disk files
- blocks records for system and blocked files
- writes volume record if BOT on multivolume magnetic tape
- performs an erase and write EOF if EOT on multivolume magnetic tape
- writes one record from the buffer pointed to by the TCW in the FCB

The base mode equivalent service is M_WRITE.

Entry Conditions

Calling Sequence

M.WRIT *fcb*

(or)

```
LA   R1,fcb
SVC  1,X'32'   (or)   M.CALL  H.IOCS,4
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO09	ILLEGAL OPERATION ON THE SYC FILE
IO38	WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE. A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF TASK HAS ONLY READ ACCESS TO UNIT.
RM02	ACCESS MODE NOT ALLOWED

Output Messages

Dismount/mount messages if EOT on multivolume magnetic tape.

6.2.140 M.XBRKR - Exit from Task Interrupt Level

The M.XBRKR service must be called after executing a task interrupt routine. This service transfers control back to the point of interruption and resets the interrupt to the level established before the break or M.INT.

The base mode equivalent service is M_XBRKR.

Entry Conditions

Calling Sequence

M.XBRKR

(or)

SVC 1,X'70' (or) M.CALL H.REXS,48

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

RX34 TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO BREAK IS ACTIVE

6.2.141 M.XIEA - No-Wait I/O End-Action Return

The M.XIEA service exits any no-wait I/O end-action routine. Both normal and error end-action routines must use this exit.

The base mode equivalent service is M_XIEA.

Entry Conditions

Calling Sequence

M.XIEA

(or)

SVC 1,X'2C' (or) M.CALL H.IOCS,34

Exit Conditions

Return Sequence

BL	S.EXEC6	no-wait I/O postprocessing complete

6.2.142 M.XMEA - Exit from Message End-Action Routine

The M.XMEA service exits the end-action routine associated with a no-wait message send request.

The base mode equivalent service is M_XMEA.

Entry Conditions

Calling Sequence

M.XMEA

(or)

SVC 1,X'7E' (or) M.CALL H.REXS,50

Exit Conditions

Return Sequence

M.RTRN return is to interrupt context at message interrupt or task base level

Abort Cases

RX99 TASK HAS MADE A MESSAGE END-ACTION ROUTINE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

6.2.143 M.XMSGR - Exit from Message Receiver

The M.XMSGR service must be called to exit the message receiver code of the calling task after the task has received a message from another task.

The base mode equivalent service is M_XMSGR.

Entry Conditions

Calling Sequence

M.XMSGR [*rxbaddr*]

(or)

```
LA   R2,rxbaddr
SVC  1,X'5E'  (or)  M.CALL   H.REXS,39
```

rxbaddr is the logical address of the receiver exit block (RXB)

Exit Conditions

Return Sequence

M.RTRN return is to interrupt context at task base level

Abort Cases

RX93 AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT

RX94 AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT

RX95 TASK HAS MADE A MESSAGE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

6.2.144 M.XREA - Exit from Run Request End-Action Routine

The M.XREA service exits the end-action routine associated with a no-wait run request.

The base mode equivalent service is M_XREA.

Entry Conditions

Calling Sequence

M.XREA

(or)

SVC 1,X'7F' (or) M.CALL H.REXS,51

Exit Conditions

Return Sequence

M.RTRN return is to interrupt context at message interrupt or task base level

Abort Cases

RX90 TASK HAS MADE A RUN REQUEST END-ACTION ROUTINE EXIT WHILE THE RUN REQUEST INTERRUPT WAS NOT ACTIVE

6.2.145 M.XRUNR - Exit Run Receiver

The M.XRUNR service exits a task that is executing for a run request issued from another task.

The base mode equivalent service is M_XRUNR.

Entry Conditions

Calling Sequence

M.XRUNR *rxbaddr*

(or)

```
LA   R2,rxbaddr
SVC  1,X'7D'  (or)  M.CALL  H.REXS,49
```

rxbaddr is the logical address of the receiver exit block (RXB)

Exit Conditions

Return Sequence

The run-receiver queue is examined. If the queue is not empty, the task is executed again on behalf of the next request. If the queue is empty, the exit options in the RXB are examined. If the option byte is zero, the task is placed in a wait state, waiting for the next run request to be received. If the option byte is nonzero, the task exits the system.

Note: If the task is re-executed, control is transferred to the instruction following the M.XRUNR call.

Abort Cases

```
RX96    AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS
        ENCOUNTERED DURING RUN RECEIVER EXIT
RX97    AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER
        ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT
RX98    TASK HAS MADE A RUN RECEIVER EXIT WHILE THE RUN
        RECEIVER INTERRUPT WAS NOT ACTIVE
```

6.2.146 M.XTIME - Task CPU Execution Time

The M.XTIME service returns to the caller the total accumulated processor execution time in microseconds since the initiation of the task. If an IPU is present and IPU accounting is enabled, the time returned includes accumulated IPU execution time, if any. If the calling task is in the real time priority range and real time accounting is turned off, the returned time will be zero.

The base mode equivalent service is M_XTIME.

Entry Conditions

Calling Sequence

M.XTIME

(or)

SVC 1,X'2D' (or) M.CALL H.REXS,65

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6,R7 CPU execution time in microseconds

Note: For a task running at real time priority (1-54) real time accounting must be on (ONRA) for the M.XTIME service to return the correct time.

6.3 Nonmacro-Callable System Services

The following system services are not in the system macro library but can be accessed by the methods indicated for each service.

6.3.1 Allocate File Space

The Allocate File Space service examines the space allocation map (SMAP) on the specified volume to locate a specified number of available contiguous allocation units. When a sufficiently long string of zero bits is located in the SMAP, the bit string is set to all ones and the SMAP is rewritten to the volume. The segment definition of the corresponding file space is returned to the caller.

The user must be privileged. This service is used in clean-up mode for volume mounting or in normal mode.

6.3.1.1 Clean-up Mode

This is indicated by bit one of R1 being set. In this mode, I/O is not performed because the SMAP is assumed in memory. The FCB should have valid values for FCB.ERWA, FCB.RECL, FCB.EQTY and FCB.ERAA. FCB.ERAA should contain the EOM block number for SMAP, for example, FCB.RECL/192W.

6.3.1.2 Normal Mode

Bit zero of the flags should be reset, the other flag bits are then ignored. The input FCB is also ignored. If a start block is specified, it is honored if possible; otherwise, an error is produced.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,19

Registers

R1	contains the FCB address and flags
R4	contains the starting block address or zero if anywhere
R5	contains the requested space size in blocks
R6	contains the MVTE address

Exit Conditions

Return Sequence

M.RTRN R4,R5

(or)

M.RTRN R7 (CC1 set)

Registers

R4,R5	contains the file segment definition
R7	contains the return status. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.3.2 Allocate Resource Descriptor

The Allocate Resource Descriptor service examines the resource descriptor allocation map (DMAP) on the specified volume to locate an available resource descriptor (RD). When a zero bit is located in the DMAP, the bit is set to 1 and the DMAP is rewritten to the volume. The disk address of the corresponding RD is returned to the caller.

If an RD is not available, the DMAP is automatically extended by allocation of available space on the volume. The DMAP is then re-examined as described above.

The user must be privileged.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,17

Registers

R1	contains the FCB address with valid FCB.ERWA and FCB.EQTY
R6	contains the MVTE address

Exit Conditions

Return Sequence

M.RTRN R4

(or)

M.RTNA R7 (CC1 set)

Registers

CC1 reset	contains the successful operation
R4	contains the RD disk address
	(or)
CC1 set	contains the error condition
R7	contains the return status. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.3.3 Create Temporary File

The Create Temporary File service creates a temporary file. The user must be privileged. The mounted volume table entry (MVTE) and starting block can be specified. This service functions as follows:

6.3.3.1 VOMM Internal Call

The file space is obtained and the RD is built in the system buffer. System FCB is held open, assigned to the RD, but the RD is not written to disk.

6.3.3.2 External Call

The file space is obtained and the RD is built in the system buffer. The RD is written to disk, and the system FCB is deassigned.

6.3.3.3 Default File Attributes

The temporary file is created with owner SYSTEM and no project group. All access is allowed, EOF management is inhibited, and the file is nonsegmented and nonextendible.

6.3.3.4 Volume Selection

If MVTE is not specified, a mounted volume is selected as follows:

Value	Description
a	the current working volume is tried first
b	the system volume is tried next
c	public mounted volumes are tried next in MVT order

Entry Conditions

Calling Sequence

M.CALL H.VOMM,24

Registers

R4	contains the starting block address or zero if anywhere
R5	contains the number of blocks required
R6	contains the MVTE or zero if anywhere

Exit Conditions

Return Sequence

CC1 set	contains the error
R4	contains the starting block
R5	contains the size in blocks
R6	contains the MVTE

The RD for the created file is in the system buffer.

R7 contains the return status. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.3.4 Deallocate File Space

The Deallocate File Space service updates the space allocation map (SMAP) on the specified volume in order to mark the specified file space as available. The user must be privileged.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,20

Registers

R4,R5	contains the file segment definition
R6	contains the MVTE address

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN R7 (CC1 set)

Registers

R7	contains return status. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.3.5 Deallocate Resource Descriptor

The Deallocate Resource Descriptor service allows privileged users to update the resource descriptor allocation map (DMAP) on the specified volume to mark the specified resource descriptor (RD) as available.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,18

Registers

R1	FCB address with valid FCB.ERWA and FCB.EQTY
R4	RD disk address
R6	MVTE address

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN R7 (CC1 set)

Registers

R7	contains return status. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

6.3.6 Debug Link Service

The Debug Link service is used only by the interactive debugger when it transfers control from the calling task to itself. The debugger places this SVC trap in the task at the specified location.

Entry Conditions

Calling Sequence

SVC 1,X'66' (or) M.CALL H.REXS,42

Exit Conditions

Return Sequence

M.RTRN

6.3.7 Eject/Purge Routine

The Eject/Purge Routine service performs the following functions:

- if a file is blocked and output active, issues a purge and returns to the user
- writes volume record if BOT is encountered on multivolume magnetic tape
- performs an erase and write EOF if EOT encountered on multivolume magnetic tape
- eject is not applicable to SYC files

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'0D'  (or)  M.CALL  H.IOCS,22
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09     ILLEGAL OPERATION ON THE SYC FILE
```

Output Messages

Mount/dismount messages if EOT is encountered on multivolume magnetic tape.

6.3.8 Erase or Punch Trailer

The Erase or Punch Trailer service writes the volume record if BOT is encountered on multivolume magnetic tape or performs an erase and write EOF if EOT is encountered on multivolume magnetic tape.

Erase or punch trailer is not applicable to blocked or SYC files.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'3E'  (or)  M.CALL   H.IOCS,21
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09     ILLEGAL OPERATION ON THE SYC FILE
```

Output Messages

Mount/dismount messages if EOT is encountered on multivolume magnetic tape.

6.3.9 Execute Channel Program

The Execute Channel Program service is available to privileged users and allows command and data chaining to General Purpose Multiplexer Controller (GPMC) and extended I/O devices only. Logical execute channel is available to both privileged and nonprivileged users. Physical execute channel is available only to privileged users.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'25' or M.CALL H.IOCS,10
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO43	INPUT/OUTPUT CONTROL LIST (IOCL) OR DATA ADDRESS NOT IN CONTIGUOUS 'E' MEMORY (GPMC DEVICES ONLY)
IO50	AN UNPRIVILEGED USER ATTEMPTED TO EXECUTE A PHYSICAL CHANNEL PROGRAM
IO51	A 'TESTSTAR' COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO52	A LOGICAL CHANNEL WAS TOO LARGE TO BE MOVED TO MEMORY POOL
IO53	A 'TIC' COMMAND FOLLOWS A 'TIC' COMMAND IN A LOGICAL CHANNEL PROGRAM
IO54	A 'TIC' COMMAND ATTEMPTED TO TRANSFER TO AN ADDRESS WHICH IS NOT WORD BOUNDED
IO55	ILLEGAL ADDRESS IN LOGICAL IOCL. ADDRESS IS NOT IN USER'S LOGICAL ADDRESS SPACE.
IO56	A READ-BACKWARD COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO57	ILLEGAL IOCL ADDRESS. IOCL MUST BE LOCATED IN THE FIRST 128K WORDS OF MEMORY.

6.3.10 Get Extended Memory Array

The Get Extended Memory Array service requests an array of extended memory. If the request cannot be met, then all free memory, except 1/8 of the amount of physical memory, is allocated to the task and a count of maps allocated is returned. This service is intended for use by tasks that require the largest possible buffers without being placed on the MRQ for an extended period.

Entry Conditions

Calling Sequence

```
LW   R1,maps
SVC  2,X'7F'   (or)   M.CALL H.MEMM,14
```

maps is the number of map blocks required

Exit Conditions

Return Sequence

M.RTRN

Registers

R2 number of map blocks allocated

R3 starting logical address of memory allocated or zero if an error occurred

R4 ending logical address of memory allocated or error code as follows:

Value	Description
1	CSECT overrun
2	request for more memory than physically exists
3	M.MEMB service in use
4	unable to allocate logically contiguous memory

R5 number of map blocks, all classes, that are now free

6.3.11 Read/Write Authorization File

The Read/Write Authorization File service is available to privileged users for performing the following functions:

- validates owner name and key and returning default information, even if key is invalid. Callable as M.CALL H.VOMM,25 only.
- validates project name and key. SVC callable by M.DEFT (H.VOMM,8).

The user must be privileged.

Entry Conditions

Calling Sequence

M.CALL H.VOMM,25

Registers

R2	zero validates owner name; one validates project name and key
R4,R5	contain the left-justified, blank-filled key, or R4 is zero and R5 contains the compressed key
R6,R7	contain the left-justified, blank-filled name

Exit Conditions

Return Sequence

Registers

R2	contains the address of an area in T.BBUFA containing the authorization entry, or zero if M.KEY file does not exist. The entry is two words for project name and one word for compressed key.
R6,R7	unchanged if the name is valid. Both R6 and R7 contain zero if the name contains invalid characters, is not in the M.KEY file, or an incorrect key is supplied.
CC1	is one if there is an unrecoverable I/O error

6.3.12 Release FHD Port

The Release FHD Port service releases the fixed head disk port reserved by the Reserve FHD Port service. This release service is available only to privileged users and is currently only supported by the four megabyte fixed head disk.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'27'   (or)   M.CALL  H.IOCS,27
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

6.3.13 Reserve FHD Port

The Reserve FHD Port service reserves a fixed head port. This service is available only to privileged tasks and is currently only supported by the four megabyte fixed head disk.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'26'   (or)   M.CALL  H.IOCS,24
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

6.4 Compatible System Services

This section contains H.ALOC, H.MONS, H.IOCS and H.FISE services that are included for compatibility with 1.*x* and 2.*x* versions of MPX-32. Using compatible services with noncompatible features, e.g., caller notification packet (CNP), is not allowed.

The MPX-32 code for the support of the H.ALOC, H.MONS, and H.FISE services can be optionally removed from the MPX-32 image by the SYSGEN NOCMS directive. Attempted execution of an H.ALOC, H.MONS, or H.FISE service with the support removed causes the calling task to abort with an SV09 abort code.

6.4.1 M.ALOC - Allocate File or Peripheral Device

The M.ALOC service dynamically allocates a peripheral device, a permanent disk file, a temporary disk file, or a SLO or SBO file, and creates a File Assignment Table (FAT) entry for the allocated unit and specified logical file code. This service may also be used to equate a new logical file code with an existing logical file code.

Entry Conditions

Calling Sequence

M.ALOC *retad,lfc,function,arga,argb,*[MOUNT],[UNBLOCKED][,WAIT]

(or)

```
LA      R1,retad
LI      R5,function
SLL     R5,24
ORMW    R5,lfc
SBR     R5,0 for MOUNT option inhibited
SBR     R5,1 for UNBLOCKED option
SBR     R5,2 for WAIT for resource requested
LA      R6,arga
LA      R7,argb
SVC     1,X'40'  (or)  M.CALL  H.MONS,21
```

retad is the denial return logical address

lfc is the 1 to 3-character ASCII logical file code to be assigned. The first byte contains zero to accommodate the function code. The LFC is then left-justified and blank-filled in the three remaining bytes.

function is the function code as follows:

Value	Description
1	assign logical file code to a user or system permanent file
2	assign logical file code to a system file code - the file must be blocked
3	assign logical file code to a peripheral device (if the device is a disk drive, a formatted volume must already be mounted on it)
4	assign logical file code to a defined logical file code - the file must be blocked
5	assign logical file code to a system permanent file only

arga and *argb* are addresses of memory locations with contents unique to each function as follows:

1	*arga*	is the 1 to 8 ASCII character permanent file name. If a user name is not associated with the calling task, the system file of the name specified is allocated. If a user name is associated with the calling task, an attempt is made to allocate a user file of the name specified. If unsuccessful, a system file is allocated.
	argb	is ignored if specified
2	*arga*	is the character string SLO or SBO in bytes 1, 2, and 3
	argb	is the number of 192-word blocks required for allocation to the file
3	*arga*	is the device type code (see Appendix A, Table A-1) in byte 0 and optionally the channel number in byte 2 and the device subaddress in byte 3. If the device subaddress is present, the most significant bit of byte 2 must be set. If the channel number is present, bit 0 of byte 0 must be set. For magnetic tape devices, byte 1 is the volume number or zero if single volume.
	argb	if *arga* defines a disk file, this is the size of file (i.e., the number of 192-word blocks required). If *arga* defines a magnetic tape, it is the 4-character reel identifier. For all other devices, is zero.
4	*arga*	is the previously defined logical file code
	argb	is zero
5	*arga*	is the 1- to 8-character permanent file name of a system file
	argb	is ignored if specified
MOUNT		specifies the mount message should not be sent
UNBLOCKED		specifies the file being allocated is to be unblocked. If not specified, the file is blocked automatically.
WAIT		specifies the caller wishes to be queued for the resource and relinquishes the CPU until the resource becomes available

Exit Conditions

Return Sequence

M.RTRN CC1 is set in the program status doubleword if the calling task has read but has not written access rights to the specified permanent file

Registers

None

(or)

Return Sequence

M.RTNA 1,6 denial returns if the requested file or device cannot be allocated

Registers

R6 equals zero if file or device is busy. Condition codes 1 to 4 are set as follows:

Code	Meaning if Set
CC1	permanent file is exclusively locked
CC2	file lock table (FLT) is full
CC3	nonshared device is already allocated
CC4	disk space is not available

equals *n* if an error condition exists as described next. When R6 equals *n*, are not applicable:

Value	Condition Codes
1	permanent file does not exist
2	reserved
3	no FAT/FPT space available
4	no blocking buffer space available
5	shared memory table entry not found
6	reserved
7	dynamic common specified in ASSIGN1
8	unrecoverable I/O error to directory
9	SGO assignment specified by terminal task
10	no UT file code exists for terminal task
11	invalid RRS entry
12	LFC in ASSIGN4 does not exit
13	assigned device not on system
14	device in use by requesting task
15	SGO or SYC assignment by real-time task
16	common memory conflicts with allocated task
17	duplicate LFC allocation attempted
18	call was incompatible

Abort Cases

MS16 TASK HAS REQUESTED DYNAMIC ALLOCATION WITH AN INVALID FUNCTION CODE

Output Messages

MOUNT messages

6.4.2 M.CDJS - Submit Job from Disk File

The M.CDJS service submits a job contained in a blocked permanent or temporary disk file. Prior to calling this service, the specified file should be rewound to purge the contents of the blocking buffer if it was dynamically built.

Entry Conditions

Calling Sequence

M.CDJS *filename*[,*password*]

(or)

```
LD   R2,password
LD   R6,filename
SVC  1,X'61'  (or)  M.CALL  H.MONS,27
```

filename is the 1- to 8-character name of the blocked permanent disk file which contains the job. If a user name is associated with the calling task, an attempt is made to allocate a user file of the name specified. If unsuccessful, a system file is allocated.

(or)

is zero in the first word and the address of a file control block associated with a blocked temporary file in the calling task in the second word.

Once submitted, the logical file code associated with the permanent or temporary file is deallocated and may be reassigned.

password is ignored if specified

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7	zero if successful
(or)	
R7	bit 0 is set if the specified file does not exist, an invalid password was specified, or the FCB is not associated with a temporary file; bits 1-31 are zero
(or)	
R7	bit 1 is one if unable to activate system input task; bits 0, 2-31 are zero

6.4.3 M.CREATE - Create Permanent File

The M.CREATE service allocates disk space for the specified permanent file and writes a corresponding entry into the specified directory. The allocated space can be zeroed.

Entry Conditions

Calling Sequence

M.CREATE *filename,blocks*,,,,[R|P],*password*,[S],[N],[F],[*type*][,Z]

(or)

```
LD    R6,filename
LW    R2,blocks
SBR   R2,2                    if N ( not SAVE DEVICE file)
SBR   R2,3                    if F ( FAST file)
ZR    R3
ZR    R4  (or) LD   R4,password
ZR    R1
LI    R1,X'type'              if type present
SBR   R1,0                    if S ( system file)
SBR   R1,1                    if Z ( prezero)
SVC   1,X'75'  (or) M.CALL  H.FISE,12
```

filename is a doubleword containing the 1 to 8 ASCII character, left-justified, blank-filled name of the file. The operating system automatically encloses the file name in single quotes; therefore, the single quote character cannot be used in a file name.

blocks is a variable word containing the size of the file specified as a multiple of 192-word blocks

[R|P] is ignored if specified

password is ignored if specified

[S] specifies the file is to be a system file. If not specified, the file is created as a user file in the current user directory.

[N] specifies the file is not to be saved in response to the SAVE DEVICE File Manager directive

[F] specifies the file is a fast file. If not entered, the file is created as a slow file.

type is a 1- or 2-digit hexadecimal value that identifies the origin of the file. File type codes are:

Value	Description
00-39	available for customer use
40-5F	reserved for system
60-9F	available for customer use
A0-AF	reserved for system
B0	base mode object file
BA	base mode shared image (or BASIC file)
BB	base mode object library file
BC	base mode macro library file
BE	base mode load module file
C0	spooled output file
CA	cataloged load module
CE	MPX-32/COFF executable image
CF	MPX-32/COFF shared image
D0	memory disk save task (J.MDSAVE) file
DB	symbolic debugger command file
ED	saved text editor file
EE	stored text editor file
FD	translated help file
FE	text editor work file
FF	SYSGEN generated file

[,Z] indicates the space allocated to the file is to be zeroed

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R7 zero if the file was not created; R6 contains the reason

R6 contains the reason the file was not created as follows:

Value	Description
1	file of the name specified already exists
2	fast file was specified and collision mapping occurred with an existing directory entry
3	reserved
4	disk space is unavailable
5	specified device (channel and/or subaddress) is not configured, or no device of the type specified is available
6	specified device is off-line
7	directory is full

Value	Description
8	specified device type (byte 1 of R3) is not configured
9	file name or password contains invalid characters or imbedded blanks
10	access mode is invalid

Abort Cases

FS01	UNRECOVERABLE I/O ERROR TO THE DIRECTORY
FS02	UNRECOVERABLE I/O ERROR TO FILE SPACE ALLOCATION MAP

6.4.4 M.DALC - Deallocate File or Peripheral Device

The M.DALC service deallocates a peripheral device or disk file to which the specified logical file code is assigned. Dynamic deallocation of a peripheral device or permanent disk file makes that resource available to other tasks. Deallocation of SLO and SBO files results in their definitions being passed to the assigned system output device. If the specified logical file code was equated to other logical file codes in the system, this service deallocates only the specified code.

Entry Conditions

Calling Sequence

M.DALC *lfc*

(or)

```
LW   R5,lfc
SVC  1,X'41'  (or)  M.CALL  H.MONS,22
```

lfc is the 1 to 3 ASCII character logical file code to be deallocated. The LFC must be left-justified and blank filled in bytes 1, 2, and 3. Byte 0 is zero unless the device is magnetic tape. If magnetic tape, bit 0 of byte 0 set to 1 specifies the dismount message is to be displayed, but the device is not deallocated; bit 0 of byte 0 reset to zero specifies the dismount message is to be displayed and the device deallocated.

Exit Conditions

Return Sequence

M.RTRN

Output Messages

**task* DISMOUNT *reel* FROM UNIT *xx*
**task* DISMOUNT *reel*, VOL *volno* FROM UNIT *xx*

task is the load module name of the task requesting the magnetic tape unit to be deallocated

reel is the reel identifier of the magnetic tape to be dismounted

volno is the volume number of the magnetic tape to be dismounted; if single volume, this field is blank

xx is the device number of the nonshared magnetic tape unit from which the tape is to be dismounted

6.4.5 M.DELETE - Delete Permanent File or Non-SYSGEN Memory Partition

The M.DELETE service deletes a permanent file or non-SYSGEN created memory partition.

Entry Conditions

Calling Sequence

M.DELETE *filename*,[S][,*password*]

(or)

```
LD   R6,filename
ZR   R3    (or)  LI   R3,G'S'
ZR   R4    (or)  LD   R4,password
SVC  1,X'77'  (or)  M.CALL H.FISE,14
```

filename is a doubleword containing the 1- to 8-character left justified, blank filled name of an existing file to be deleted

[S] specifies a system file is to be deleted. If not specified, the file is assumed to be a user file whose user name is that which is associated with the calling task.

[,*password*] is ignored if specified

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6,R7 unchanged if the deletion was successful

(or)

R6 contains the reason that the specified file was not deleted:

Value	Description
1	file of the name specified does not exist or is a SYSGEN created memory partition, or the file is allocated, or no FLT space is available
2	invalid
3	access rights

R7 zero if the specified file was not deleted

Abort Cases

```
FS01    UNRECOVERABLE I/O ERROR TO THE DIRECTORY
FS02    UNRECOVERABLE I/O ERROR TO FILE SPACE ALLOCATION
        MAP
```

6.4.6 M.EXCL - Free Shared Memory

The M.EXCL service allows a task to dynamically deallocate any common areas it has previously shared using the M.SHARE service or included M.INCL service.

Entry Conditions

Calling Sequence

M.EXCL *partition*{,*ownername*|,*taskno*,TNUM}

(or)

```
LD   R6,partition
LD   R2,ownername  } (or)        LW   R2,taskno
                                 ZR   R3
SVC  1,X'79'   (or)  M.CALL  H.ALOC,14
```

partition	is the name of the partition to be deallocated. The name must be doubleword bounded, left-justified and blank-filled, such as GLOBAL01
ownername	is the owner name of the original owner of the partition
taskno	is the left-justified task number of the original owner of the partition
TNUM	specifies a task number is being used instead of an owner name

Exit Conditions

Return Sequence

M.RTRN (or) abort user with AL39

Abort Cases

```
AL39     SHARED MEMORY ENTRY NOT FOUND
```

6.4.7 M.FADD - Permanent File Address Inquiry

The M.FADD service issues I/O directly. It provides the word address of the beginning of a memory partition or the track, head, and sector address of the beginning of a disk file. The device address for disk files is also included in the result. Access restrictions are returned for disk files.

Entry Conditions

Calling Sequence

M.FADD *name*

(or)

```
LD   R6,name
SVC  1,X'43'  (or)  M.CALL  H.MONS,2
```

name is a doubleword containing the 1- to 8-character ASCII, left-justified, blank-filled permanent file name or partition name. If a file name is specified, an attempt is made to locate the file in the current working directory associated with the calling task. If the file is not found, an attempt is made to locate the file in the system directory.

Exit Conditions for Case I, Denial Return

Return Sequence

M.RTRN 7

Registers

R7 bit 0 is set to one indicating that the specified file name cannot be located; bits 1-31 are zero

Exit Conditions for Case II, Memory Partition

Return Sequence

M.RTRN 6,7

Registers

R6 bit 0 is set to one indicating that the specified name is a memory partition; bits 1-31 are the number of 512-word pages allocated to partition

R7 contains the logical address of the first word of the specified partition

Exit Conditions for Case III, Disk File

Return Sequence

M.RTRN 6,7

Registers

R6 and R7 are returned with the address of the beginning of the disk file as follows:

0 – 5	6 – 12	13 – 15	16 – 31
Zero	Device address	Zero	Track Number

0	1 – 7	8 – 15	16 – 23	24 – 31
0	Channel address	Device subaddress	Head number	Sector number

The device address is the sum of the channel address and device subaddress. It is positioned so that it may be ORed into a CD instruction (for nonmultiplexed and nonextended devices).

The following additional parameters are returned:

- CC1 is set in the program status word if a password is required to write a read-only file.
- CC2 is set if a password is required to read or write a password-only file.
- CC3 is set if the file is a system file or a core partition.

6.4.8 M.FILE - Open File

The M.FILE service performs the following functions:

- establishes appropriate linkages between a user FCB and an assigned file or device
- marks a file open for either update access or read-only operations. Increments internal counts of tasks having the file open at this time.
- for SYC or SGO files, completes building the FAT based on job control information
- for system and blocked files, initializes the blocking buffer for subsequent accesss
- requests the initial mount message for statically allocated, nonshared magnetic tape devices

Open requests to a file which is already opened are ignored.

Note: This system service is not excluded by the SYSGEN NOCMS directive.

Entry Conditions

Calling Sequence

M.FILE *fcb*[,RW]

(or)

```
LA     R1,fcb
[SBR   R1,1 if RW]
SVC    1,X'30'  (or)  M.CALL H.IOCS,1
```

fcb	is the FCB address
[,RW]	specifies update access to the file. If RW is not specified or if update access is not compatible with existing access rights to the file, the file is opened read-only.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO20    AN ERROR HAS OCCURRED IN THE REMM OPEN PROCEDURE
RM29    INVALID FILE CONTROL BLOCK (FCB) ADDRESS OR
        UNASSIGNED LFC IN FCB
```

6.4.9 M.FSLR - Release Synchronization File Lock

The M.FSLR service is used for disk file gating. It is implemented with the Set Synchronization File Lock service (M.FSLS) to control a synchronization lock indicator. When M.FSLR is called, the synchronization lock is released, and the queue of tasks waiting to own the lock is polled.

A synchronization lock can only be cleared by the task that set the lock.

A synchronization lock is automatically released when the owner task terminates the file is deallocated.

The locked file is deallocated.

Entry Conditions

Calling Sequence

M.FSLR *lfca*

(or)

```
LW   R5,lfca
SVC  1,X'24'  (or)  M.CALL  H.FISE,25
```

lfca is the address of a word that contains an unused byte in byte 0, and a 1- to 3-character ASCII, left-justified, blank-filled logical file code in bytes 1, 2, and 3

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 Value	Description
0	request accepted, synchronization lock released
1	request denied, synchronization lock was not set
5	request denied, specified LFC not allocated
6	request denied, specified LFC is not assigned to a permanent disk file

6.4.10 M.FSLS - Set Synchronization File Lock

The M.FSLS service is used with the Release Synchronization File Lock service (M.FSLR) for disk file gating. The M.FSLS and M.FSLR services control a synchronization lock indicator that allows synchronized access to a disk file concurrently allocated to multiple tasks. To use the M.FSLS service, the file must have been previously allocated to the calling task.

Entry Conditions

Calling Sequence

M.FSLS *lfca*[,*timev*]

(or)

```
LW   R5,lfca
LI   R4,timev  (or)  ZR  R4
SVC  1,X'23'   (or)  M.CALL H.FISE,24
```

lfca is the address of a word that contains an unused byte in byte 0, and a 1- to 3-character ASCII, left-justified, blank-filled logical file code in bytes 1, 2, and 3

timev is a numeric value interpreted as follows:

Value	Description
+1	return immediately with a denial code if the file already has a synchronization lock set
0	place the requesting task in a wait state until it has become the owner of the synchronization lock
-*n*	place the requesting task in a wait state until it owns the synchronization lock, or until the expiration of *n* timer units, whichever occurs first

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 Value	Description
0	request accepted, synchronization lock set
1	request denied, synchronization lock is already owned by another task
2	request denied, time out occurred while waiting to become lock owner
3	request denied, matching FLT entry not found
4	reserved
5	request denied, LFC not assigned to permanent disk file

6.4.11 M.FXLR - Release Exclusive File Lock

The M.FXLR service is used in conjunction with the set exclusive file lock service (M.FXLS) for disk file gating. When M.FXLR is called, the exclusive lock is released and other tasks can allocate the associated disk file.

An exclusive file lock can not be released by a task other than the owning task. Therefore, another task cannot exclusively lock the file until it is deallocated by the owning task.

Any outstanding exclusive file locks are released on task termination or on file deallocation.

Entry Conditions

Calling Sequence

M.FXLR *lfca*

(or)

```
LW   R5,lfca
SVC  1,X'22'  (or)  M.CALL   H.FISE,23
```

lfca is the address of a word that contains an unused byte in byte 0, and a 1- to 3-character ASCII, left-justified, blank-filled logical file code in bytes 1, 2, and 3

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 Value	Description
0	request accepted, exclusive file lock released
29	request denied, specified LFC is not assigned by this task
30	request denied, invalid allocation index
32	request denied, an exclusive resource lock was not owned by this task
33	request denied, resource is not allocated in a shareable mode by this task

6.4.12 M.FXLS - Set Exclusive File Lock

The M.FXLS service is used for disk file gating. It allows the calling task to gain exclusive allocation of a file, as though the file were an unshared resource. The file must have been previously allocated, and is identified by the address of logical file code (LFC).

Entry Conditions

Calling Sequence

M.FXLS *lfca*[,*timev*]

(or)

```
LW   R5,lfca
LI   R4,timev  (or)  ZR  R4
SVC  1,X'21'   (or)  M.CALL  H.FISE,22
```

lfca is the address of a word that contains an unused byte in byte 0, and a 1- to 3-character ASCII, left-justified, blank-filled logical file code in bytes 1, 2, and 3

timev is a numeric value interpreted as follows:

Value	Description
+1	return immediately with a denial code if the file is already allocated to another task
0	place the requesting task in a wait state until the designated file can be exclusively locked
-*n*	place the requesting task in a wait state until the designated file can be exclusively locked, or until the expiration of *n* timer units, whichever occurs first

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 Value	Description
0	request accepted, file is exclusively locked
1	request denied, file is allocated to another task, or is already exclusively locked
2	reserved
3	reserved
4	request denied, time out occurred while waiting to become lock owner
6	request denied, LFC not assigned to permanent disk file

6.4.13 M.INCL - Get Shared Memory

The M.INCL service allows a task to dynamically include a memory partition into its address space, e.g., GLOBAL01 or DATAPOOL common. The task is suspended until the inclusion is complete. The calling task that performs an M.INCL specifies the owner name or task number, whichever was entered into the shared memory table.

Entry Conditions

Calling Sequence

M.INCL *partition*{,*ownername* | ,*taskno*},[RW],[*password*],[*denial*][,TNUM]

(or)

```
LD   R6,partition
LD   R2,ownername  } (or)        LW   R2,taskno
                                 ZR   R3
ZR   R0
ZR   R4    (or) LD  R4,password
ZR   R5
LA   R0,denial
[SBR  R0,0 if RW]
SVC  1,X'72'   (or)   M.CALL  H.ALOC,13
```

partition	is the doubleword-bounded, left-justified, blank-filled memory partition name, such as GLOBAL01
ownername	is the owner name of the original owner of the partition
taskno	is the left-justified task number of the original owner of the partition
[RW]	specifies read and write control desired
[*password*]	is ignored if specified
[*denial*]	is a denial return address
[,TNUM]	specifies a task number is being used instead of an owner name

Exit Conditions

Return Sequence

M.RTRN R3

Registers

R3	contains the starting address of the shared memory partition. This is a 20-bit address.
	(or)

M.RTNA R0,R3 for denial returns

R3	Value	Description
	1	entry not found in shared memory table
	2	reserved
	3	memory requirements conflict with task's address space
	4	entry not found in shared memory table after returning from SWGQ state chain

R0 address to return to within user task body

6.4.14 M.LOG - Permanent File Log

The M.LOG service provides a log of currently existing permanent files.

Entry Conditions

Calling Sequence

M.LOG *type,address*[*,filename*]

(or)

```
LI    R4,type
LA    R5,address
LD    R6,filename (if TYPE =N or O)
SVC   1,X'73'  (or)  M.CALL  H.MONS,33
```

type is a byte-scaled value which specifies the type of log to be performed as follows:

Value	Description
=N=0	specifies a single named file in the current working directory or system directory
=A=1	specifies all permanent files in the current working directory
=S=2	specifies all permanent files in the system directory
=U=3	specifies all permanent files in the current working directory
=O=4	specifies a single named file in the system directory

If type is N, an attempt is made to locate the file in the current working directory associated with the calling task. If the file is not found, an attempt is made to locate the file in the system directory.

address is the address of an 8-word area within the calling task where a copy of a SMD entry is to be stored

[*,filename*] is a 1- to 8-character file name if *type* equals N or O

Exit Conditions

Return Sequence

M.RTRN 4,5

An 8-word SMD entry, if any, is stored at the address specified in *address*. The password field contains zero to indicate the absence of a password.

Registers

R4 is zero if *type* is N or O. For type A, S, or U where the service is called repeatedly, the type is specified only on the first call. R4 contains the address of the next directory entry to be returned. R4 value must be unchanged on subsystem calls to this service.

R5 contains zero if *type* equals N or O (R4 equals zero or four) or *type* is A, S, or U (R4 equals one, two, or three) and file could not be found. Otherwise, R5 is unchanged.

Abort Cases

MS28 A PERMANENT FILE LOG HAS BEEN REQUESTED, BUT THE ADDRESS SPECIFIED FOR STORAGE OF THE DIRECTORY ENTRY IS NOT CONTAINED WITHIN THE CALLING TASK'S LOGICAL ADDRESS SPACE

Note: The M.LOG system service searches the memory resident descriptor table (MDT) for resource descriptors before it searches the disk-resident resource descriptors. For MDT information, refer to the Rapid File Allocation Utility (J.MDTI) chapter in Volume II.

6.4.15 M.PDEV - Physical Device Inquiry

The M.PDEV service returns physical device information describing the unit to which a specified logical file code is assigned.

Entry Conditions

Calling Sequence

M.PDEV *lfc*

(or)

```
LW   R5,lfc
SVC  1,X'42'   (or)   M.CALL  H.MONS,1
```

lfc is a 1 to 3 ASCII character LFC, left-justified and blank filled, in bytes 1 to 3. For a system FAT/FPT pair, bit 0 of byte is set.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if the specified logical file code is unassigned

(or)

Return Sequence

M.RTRN 4,5,6,7

Registers

R4 bit 0 is one for extended I/O (Class F) device; zero for all other device classes
bits 1-7 are zero
byte 1 is zero
bytes 2 and 3 are 2-character ASCII device codes such as MT, DC, etc.)

R5 if disk, contains the number of 192-word blocks in file. If magnetic tape, contains the 4-character ASCII reel identifier. For TSM terminal:

Byte	Contents
2	number of hexadecimal characters in line
3	number of hexadecimal lines on screen

All other devices equal 0.

R6

Bytes 0 and 1	maximum number of bytes transferrable to device
Byte 2	device channel number
Byte 3	device subaddress

R7 Byte 0 device type code as two hexadecimal digits. See Appendix A.

Byte 1

Bit 8 zero if file is unblocked; one if file is blocked

Bits 9-12 zero

Bits 13-15 system file code, as follows:

Code	Description
0	not a system file
1	SYC file
2	SGO file
3	SLO file
4	SBO file

Bytes 2 and 3 for disk, the number of 192-word sectors per allocation unit; for magnetic tape, the volume number or zero for a single volume; for all other devices, zero

If the specified logical file code is assigned to SYC or SGO, and that file is not open, bits 13 through 15 of R7 are returned as equal to one or two. No other returned parameters are applicable.

When inquiring about physical device from the console terminal, R7 can be returned equal to zero even though the LFC is assigned. When this occurs, the device type code 00 (console terminal) is in R7.

6.4.16 M.PERM - Change Temporary File to Permanent

The M.PERM service changes the status of a temporary file allocated to the calling task to permanent.

Entry Conditions

Calling Sequence

M.PERM *filename,lfc*[,[{R|P},*password*],[S],[N],[F],[*type*][,Z]]

(or)

```
LD      R6,filename
LW      R2,=G'lfc'
ZR      R3
[SBR    R3,6                    if R - read only
 SBR    R3,7                    if P - password only
 SBR    R3,2                    if N - not SAVE DEVICE file
 SBR    R3,3                    if F - fast file ]
ZR      R1
[LI     R1,X'type' if type present
 SBR    R1,0                    if S - system file
 SBR    R1,1                    if Z - pre zero ]
ZR      R4
[LD     R4,password if file is to have a password ]
 SVC    1,X'76'  (or)  M.CALL  H.FISE,13
```

filename is a doubleword containing the 1- to 8-character ASCII, left-justified, blank-filled name of the file. The operating system automatically encloses the file name in single quotes; therefore, the single quote character cannot be used in a filename.

lfc is the 1- to 3-character ASCII, right-justified, zero-filled logical file code assigned to an open, temporary SLO, or SBO file. The file is marked as permanent in the calling task by this service if successful.

[{R,P},*password*] are ignored if specified

[S] specifies the file is to be a system file. If not specified, the file is created as a user file if a user name is associated with the calling task, or as a system file if no user name is associated with the calling task.

[N] is ignored if specified

[F] specifies the file is a fast file. If not specified, the file is created as a slow file.

[*type*] is a 2-digit hexadecimal value that identifies the origin of the file. File types codes are:

Value	Description
00-39	available for customer use
40-5F	reserved for system
60-9F	available for customer use
A0-AF	reserved for system

Value	Description
B0	base mode object file
BA	base mode shared image (or BASIC file)
BB	base mode object library file
BC	base mode macro library file
BE	base mode load module file
C0	spooled output file
CA	cataloged load module
CE	MPX-32/COFF executable image
CF	MPX-32/COFF shared image
D0	memory disk save task (J.MDSAVE) file
DB	symbolic debugger command file
ED	saved text editor file
EE	stored text editor file
FD	translated help file
FE	text editor work file
FF	SYSGEN generated file

[,Z] is ignored if specified

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R7 is unchanged if successful; otherwise is zero if the file was not created

R6 contains the reason that the file was not created:

Value	Description
1	file of the name specified already exists
2	fast file was specified and collision mapping occurred with an existing directory entry
3	restricted access but no password supplied
4	file associated with the specified logical file code is not a temporary file, not open, not SLO or SBO, or no FLT space
7	directory is full
9	file name contains invalid characters

Abort Cases

```
FS01    UNRECOVERABLE I/O ERROR TO THE DIRECTORY
FS02    UNRECOVERABLE I/O ERROR TO FILE SPACE ALLOCATION
        MAP
```

6.4.17 M.SHARE - Share Memory with Another Task

The M.SHARE service dynamically creates a shared memory partition from the partition definition in the system directory. This definition must have been previously defined using the currently unsupported File Manager utility.

The call results in the creation of a new common area, which is uniquely identified by the owner name or task number of the caller, and by the memory partition name. The memory type is specified by the directory definition. Prezeroing is not performed by this service. The partition is swappable with the task if the use count equals zero. The partition is deallocated when the allocation count equals zero. The task is suspended until the shared memory table entry is built and the memory allocation is complete. The shared partition can be gated from use by other tasks to allow the initial loading of data. This is called a data lock. Any tasks attempting to include this partition while the lock is set are queued to the SWGQ state (general queue) and remain there until the lock is reset by the M.SMULK service.

Options are:

- request read and write access, set bit 0 in R0
- request task number instead of owner name, set bit 1 in R0
- request data lock and includers to be enqueued, set bit 2 in R0

Entry Conditions

Calling Sequence

M.SHARE *partition*,[RW],[*password*],[TNUM][,LOCK]

(or)

```
LD    R6,partition
ZR    R0
SBR   R0,0 if read/write
SBR   R0,1 if task number requested
SBR   R0,2 if data lock requested
LD    R4,password
ZR    R4 if no password
ZR    R5 if no password
SVC   1,X'71'   (or)  M.CALL  H.ALOC,12
```

partition	is the doubleword-bounded, left-justified memory partition name
[RW]	is ignored if specified
[*password*]	is ignored if specified
[TNUM]	specifies a task number is being used instead of an owner name
[,LOCK]	specifies a data lock

Exit Conditions

Return Sequence

M.RTRN R3 (or) abort user with AL40 or AL41

Registers

R0	bit 4 is reset if share becomes an include
R3	contains the starting address of the memory partition if share is successful

Abort Cases

AL40	PARTITION DEFINITION NOT FOUND IN DIRECTORY
AL41	DIRECTORY DEFINITION NOT A DYNAMIC PARTITION

6.4.18 M.SMULK - Unlock and Dequeue Shared Memory

The M.SMULK service unlocks the data lock associated with a particular shared memory partition. See M.SHARE (ALOC,12) for use of data lock.

After M.SMULK is executed, the lock on the shared area is reset and all users queued to the shared area are relinked from the SWGQ (general queue wait state) to their appropriate run state. They then have full access to the shared partition.

Entry Conditions

Calling Sequence

M.SMULK *partition,ownername*[,TNUM]

(or)

```
LD   R2,ownername      (or)      { LW R2,taskno
                                 { ZR R3
SVC  1,X'1F'   (or)  M.CALL  H.ALOC,19
```

partition	is a left-justified, blank-filled, doubleword bounded memory partition name
ownername	is the owner name of the original owner of the partition
taskno	is the left-justified task number of the original owner of the partition
[,TNUM]	specifies a task number is being used instead of an owner name

Exit Conditions

Return Sequence

M.RTRN

6.4.19 M.USER - User Name Specification

The M.USER service associates a user name with the calling task. This service can nullify any user name associated with the calling task. The user name associated with the task is used in file create, delete, log, and allocate services called subsequently.

Entry Conditions

Calling Sequence

M.USER [*username*][,*key*]

(or)

```
ZR   R6 null username
ZR   R7
SVC  1,X'74'  (or)  M.CALL  H.MONS,34
```

(or)

```
LD   R6,username
LD   R4,key
SVC  1,X'74'  (or)  M.CALL H.MONS,34
```

[*username*] is the 1- to 8-character user name that is left-justified and blank-filled. Each character must have an ASCII equivalent in the range 01 through 7F.

[,*key*] is ignored if specified

If both parameters are omitted, *username* defaults to system on the current working volume.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6,R7 contain zero if the service was not performed because the specified user name contains invalid characters or is not in the user name file or the required key was not furnished; otherwise, unchanged

7 Base Mode System Services

7.1 General Description

MPX-32 resident base mode system service routines perform frequently required operations with maximum efficiency by using the Supervisor Call (SVC) instruction. Tasks running in any environment can call these routines.

All system service routines are reentrant. Thus, each service routine is always available to the task that is currently active.

System service routines are provided as standard modular components of MPX-32. The open-ended design of the system, however, allows service routines to be added to tailor MPX-32 to a specific application.

System services enable tasks to:

- activate, suspend, resume, abort, terminate and hold task execution
- change a task's priority level
- create, test, and delete timers
- interrogate system clocks
- allocate and deallocate devices and files
- obtain the characteristics of a device or file
- communicate with other tasks through messages and status words
- load and execute overlays
- obtain information about the memory assigned to a task
- connect tasks to interrupts
- determine the arithmetic exception and option word status for a task

MPX-32 services are implemented as SVC traps. There are several ways of accessing services:

1. By macro calls, with parameter passing as indicated in the individual descriptions. The expansion code in the system macro library is then accessed automatically during assembly to provide Assembly language setup of appropriate registers and instructions, including SVCs, in user code.
2. By setting up appropriate registers and instructions directly and using appropriate SVCs.
3. By following number 2 above but issuing an M_CALL request to the entry point of the system module that provides the service.

The first two access paths are described for each system service. The third access path is privileged, and is indicated primarily to provide the appropriate system module names and entry point numbers for cross-reference to other documentation when needed.

Special operations performed for a task are:

- Open — If not issued by the task, IOCS opens the file or device for the default access in effect at that time.
- Close — If not issued by the task, the file is closed automatically and a device is deallocated automatically during task termination.

Callable system services are described in alphabetical order by macro name. Available system services that are not macro callable are described in Section 7.3.

7.1.1 Syntax Rules and Descriptions

System services can be called by their macro name, their SVC number, or their module entry point number. It is recommended that whenever possible the macro name be used. When a macro name is used, any optional parameter not specified in the call is handled as follows:

- the appropriate register is assumed to have been previously loaded

(or)

- the appropriate register will be zeroed

Refer to the calling sequence description of each service to determine applicability of missing parameter handling.

Defaults for optional parameters are documented in the description of each service.

When a required parameter is not specified or an invalid parameter is specified, an error message is displayed in the listing regardless of the listing controls in effect.

The integrity of the condition code setting on exit is not guaranteed for system services, except as documented for a particular service. Refer to the description of each service to determine whether the exit condition code settings are applicable.

Base mode system services can only be used with the Macro Assembler/X32. When using base mode system services, expanded parameter specification rules apply.

7.1.1.1 Parameter Specification

Parameters can be specified to a base mode service in two ways: by keyword and by position.

When parameters are specified by keyword, the parameters are not order dependent. Each parameter is denoted by using the parameter keyword followed by an equal sign followed by the value. Multiple parameter groups are separated by a comma. Refer to Example 1.

When parameters are specified by position, only the value of a parameter is specified. The parameters are order dependent and must be entered in the order shown in the syntax of each service. Multiple parameter groups are separated by a comma. A comma must also be inserted for an optional parameter not specified if a following optional parameter is specified in the syntax. Refer to Example 2.

Keyword and positional parameters can be mixed within a syntax statement. However, mixing is not recommended since the position is not advanced when keyword parameters are processed by the assembler.

Example 1 — Specifying Parameters by Keyword

To create a permanent file using the M_CREATEP service, any one of the following keyword syntax variations are valid:

M_CREATEP PNADDR=*addr1*,RCBADDR=*addr2*,CNPADDR=*addr3*
M_CREATEP PNADDR=*addr1*,RCBADDR=*addr2*
M_CREATEP PNADDR=*addr1*,CNPADDR=*addr3*
M_CREATEP RCBADDR=*addr2*,CNPADDR=*addr3*,PNADDR=*addr1*

Example 2 — Specifying Parameters by Position

To create a permanent file using the M_CREATEP service, any one of the following positional syntax variations are valid:

M_CREATEP *addr1*,*addr2*,*addr3*
M_CREATEP *addr1*,*addr2*
M_CREATEP *addr1*,,*addr3*
M_CREATEP *addr1*

Parameter Value Specification

Parameter values can be specified in the following ways:

- by a register name
- by a base register name
- by a label that refers to a required data structure
- by a label that refers to a memory location containing the address of a required data structure
- by a literal
- by defaults

When using a register name R0 through R7, the value to be used is in the specified register. For example,

```
M_GETMEMBYTES R6
```

generates:

```
TRR  R6,R4          !load correct register
SVC  2,X'4B'        !enter service
```

When using a base register name B0 through B7, the value to be used is in the specified base register. Specifying base registers B0 through B3 may cause errors, as the base mode operating system uses those registers for call and return instructions. For example,

```
M_GETMEMBYTES B6
```

generates:

```
TBRR  B6,R4         !load correct register
SVC   2,X'4B'       !enter service
```

When using a label that refers to a memory location, the associated register is automatically selected. For example,

```
M_OPENR FCBOUT
```

generates:

```
LW   R1,#A'FCBOUT'  !get FCB address
SVC  2,X'42'        !enter service
```

When using a label that refers to an absolute memory location and an explicitly defined base register, the sum of the address of the label relative to the start of the program segment and the contents of the register equal the destination address. For example,

```
M_OPENR @FCBLOC(B3)
```

generates:

```
LW   R1,FCBLOC(B3)  !get FCB address
SVC  2,X'42'        !enter service
```

When using a literal, the literal must be preceded by a # symbol. For example,

```
M_GETMEMBYTES #8192
```

generates:

```
LW   R4,#8192       !get byte count
SVC  2,X'4B'        !enter service
```

When using defaults, the value passed to the service is the default value defined for the missing parameter. For example,

```
M_AWAITACTION
```

generates:

```
LW   R6,?0          !ZERO is a location containing zero
SVC  1,X'1D'        !enter service
```

7.1.2 IPU Executable Base Mode System Services

Once a task has gained entry into the IPU, there is a limited set of system services that the IPU can execute. These are memory reference only system services, since the IPU cannot execute any I/O instructions. The following base mode system services are executable in the IPU:

SVC	Description
M_ADRS	Memory Address Inquiry
M_BBTIM	Acquire Current Date/Time in Byte Binary Format
M_BTIM	Acquire Current Date/Time in Binary Format
M_CMD	Get Command Line
M_CONABB	Convert ASCII Date/Time to Byte Binary Format
M_CONADB	Convert ASCII Decimal to Binary
M_CONAHB	Convert ASCII Hexadecimal to Binary
M_CONASB	Convert ASCII Date/Time to Standard Binary
M_CONBAD	Convert Binary to ASCII Decimal
M_CONBAF	Convert Binary Date/Time to ASCII Format
M_CONBAH	Convert Binary to ASCII Hexadecimal
M_CONBBA	Convert Byte Binary Date/Time to ASCII
M_CONBBY	Convert Binary Date/Time to Byte Binary
M_CONBYB	Convert Byte Binary Date/Time to Binary
M_CTIM	Convert System Date/Time Format
M_CONVERTTIME	Convert Time
M_DATE	Date and Time Inquiry
M_DEVID	Get Device Mnemonic or Type
M_DSMI	Disable Message Task Interrupt
M_DSUB	Disable User Break Interrupt
M_ENUB	Enable User Break Interrupt
M_ENVRMT	Get Task Environment
M_GETTIME	Get Current Date and Time
M_GTIM	Acquire System Date and Time in any Format
M_GTSAD	Get TSA Start Address
M_OPTIONDWORD	Task Option Doubleword Inquiry
M_OPTIONWORD	Task Option Word Inquiry
M_OSREAD	Physical Memory Read
M_OSWRIT	Physical Memory Write
M_QATIM	Acquire Current Date/Time in ASCII Format
M_SYNCH	Set Synchronous Task Interrupts
M_TDAY	Time-of-Day Inquiry
M_TSTE	Arithmetic Exception Inquiry
M_TSTT	Test Timer Entry

7.2 Macro-Callable System Services

All base mode system services are described in detail in the following pages. System services that are supported for base mode tasks begin with "M_".

7.2.1 M_ACTV - Activate Task

The M_ACTV service activates a task. The task assumes the owner name of the caller. When a load module is supplied as input, the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied as input.

The nonbase mode equivalent service is M.ACTV.

Entry Conditions

Calling Sequence

M_ACTV [LOADMOD=]*loadmod*

(or)

```
LD   R6,loadmod
SVC  1,X'52'  (or)  M_CALL   H.REXS,15
```

loadmod is either a left-justified blank-filled doubleword containing the 1- to 8-character ASCII name of the load module for which an activation request is queued (must be a system file), or R6 is zero and R7 contains the pathname vector or RID vector which points to the load module to be activated

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 equals zero if the service could be performed
R7 contains the task number of the task activated by this service

(or)

R6 equals one if invalid attempt to multicopy a unique load module
R7 task number of existing task with same name

(or)

R6 Value	Description
2	if load module file not in directory
3	unable to allocate load module
4	if file is not a valid load module
5	if DQE is not available
6	if read error on resource descriptor
7	if read error on load module
8	insufficient logical/physical address space for task activation

7.2.2 M_ADRS - Memory Address Inquiry

The M_ADRS service provides the beginning and ending logical addresses of the memory allocated to a task. The beginning address is the location into which the first word was loaded and is a word address. The ending address is also a word address and defines the last word allocated to the task.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.ADRS.

Entry Conditions

Calling Sequence

M_ADRS

(or)

SVC 1,X'44' (or) M_CALL H.REXS,3

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	logical word address of the first location of the task's DSECT. This address is always on a page boundary.
R7	logical word address of the last location available for loading or expansion of the task's DSECT. This address is always on a map block boundary minus one word.

7.2.3 M_ADVANCE - Advance Record or File

The M_ADVANCE service performs the following functions for advancing records:

- verifies volume record if BOT is encountered on multivolume magnetic tape
- advances specified number of records

M_ADVANCE performs the following functions for advancing files:

- if the file is blocked, logical records are advanced until an end-of-file is found. The read/write control word will point to the first record after the end-of-file.
- verifies volume record if BOT is encountered on multivolume magnetic tape
- Advances specified number of files

The M_ADVANCE service cannot be used for SYC files or unblocked disk files.

The nonbase mode equivalent service is M.FWRD.

Entry Conditions

Calling Sequence

M_ADVANCE [FCBADDR=]*fcbaddr*,[[MODE=]{R | F}] [NUMBER=]*number*

(or)

```
SVC     1,X'33'  (or)  M_CALL H.IOCS,7   } (or)
SVC     1,X'34  (or)  M_CALL H.IOCS,8    }
BIB     R4,$-1W
```

fcbaddr	is the FCB address
R	specifies advance by record (SVC 1,X'33')
F	specifies advance by file (SVC 1,X'34'); F is the default
number	is the address of the word containing the number of records or files to be advanced, or a value of one, if not specified
$-W	branches back to SVC the number of times specified by *number*

Registers

R1	contains *fcbaddr*
R4	contains *number*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO07	THE TASK HAS ATTEMPTED TO PERFORM AN OPERATION WHICH IS NOT VALID FOR THE DEVICE TO WHICH THE USER'S FILE IS ASSIGNED (E.G., A READ OPERATION SPECIFIED FOR A FILE ASSIGNED TO THE LINE PRINTER)
IO09	ILLEGAL OPERATION ON THE SYC FILE
IO30	ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE

Output Messages

Mount/dismount messages if EOT encountered on multivolume magnetic tape.

7.2.4 M_ANYWAIT - Wait for Any No-Wait Operation Complete, Message Interrupt, or Break Interrupt

The M_ANYWAIT service places the currently executing task in a state waiting for the completion of any no-wait request, for the receipt of a message, or for a break interrupt. The task is removed from the associated ready-to-run list, and placed in the any-wait list. A return is made to the program location following the SVC instruction only when one of the wait conditions has been satisfied or when the optional time-out value has expired.

The nonbase mode equivalent service is M.ANYW.

Entry Conditions

Calling Sequence

M_ANYWAIT [[WAITTIME=]*time1*]

(or)

```
LW   R6,time1
SVC  1,X'7C'   (or)   M_CALL   H.REXS,37
```

time1 contains zero if wait for an indefinite period is requested; otherwise, *time1* contains the negative number of time units to elapse before the wait is terminated

Registers

R6 contains *time1*; otherwise, zero

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

MS31 USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN END-ACTION ROUTINE

7.2.5 M_ASSIGN - Assign and Allocate Resource

The M_ASSIGN service associates a resource with a logical file code (LFC) used by a process and allocates the resource. This function creates a file assignment table/file pointer table (FAT/FPT) pair within the user's task service area (TSA) and associates an allocated resource table (ART) entry for system administration and control of the resource while allocated. When implicit sharing is indicated by the absence of a specified usage mode, the appropriate linkage is established to coordinate concurrent access. The option is provided to allocate and open a resource with a single call to this function.

The nonbase mode equivalent service is M.ASSN.

Entry Conditions

Calling Sequence

M_ASSIGN [RRSADDR=]*addr* [,[CNPADDR=]*cnpaddr*]

(or)

```
LA   R1,addr
LA   R7,cnpaddr      (or)   ZR   R7
SVC  2,X'52'   (or)   M_CALL   H.REXS,21
```

addr is the address of a resource requirement summary (RRS) entry, (type 1 through 6)

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are time-out value, abnormal return address, option field, status field, and parameter link.

The option field contains an access and usage specification for opening this resource. This field is used only if the automatic open flag is set in the option word of the RRS. See the M_OPENR service.

If automatic open is indicated in the resource requirement summary, word 5 of the CNP must contain the address of a valid file control block (FCB) for this assignment. See the M_OPENR service.

Registers

R1 contains *addr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R5	M.RTRN R5
(or)	(or)
M.RTRN R5 (CC1 set)	M.RTRN R5,R7 (CC1 set)

Registers

R5 contains the allocation index, a unique 32-bit integer number associated with the allocated resource. This index may be used to set and release resource locks for exclusive or synchronous access.

R7 contains return status if a CNP is not supplied, otherwise, unchanged

Status

CC1 set

The following values are posted in R7 or the status field of the CNP. Status values 25-29 are returned only when auto-open is indicated.

Value	Description
1	unable to locate resource because of invalid pathname
2	specified access mode not allowed
3	FAT/FPT space not available
4	blocking buffer space not available
7	static assignment to dynamic common
8	unrecoverable I/O error to volume
9	invalid usage specification
11	invalid RRS entry
12	LFC logically equated to unassigned LFC
13	assigned device not in system
14	resource already allocated by requesting task
15	SGO or SYC assignment by real-time task
17	duplicate LFC assignment attempted
19	invalid resource ID
20	specified volume not assigned
22	resource is marked for deletion
23	assigned device is marked offline
24	segment definition allocation by unprivileged task
25	random access not allowed for this access mode
27	resource already opened in a different access mode
28	invalid access specification at open

Value	Description
29	specified LFC is not assigned to a resource for this task
38	time out occurred while waiting for resource to become available
46	unable to obtain resource descriptor lock available to multiprocessor only
50	resource is locked by another task
51	shareable resource is allocated in an incompatible access mode
54	unable to allocate resource for specified usage
55	allocated resource table (ART) space not available

Wait Conditions

When the resource is not available, as indicated by status values 50-63, the task is placed in a wait state, as appropriate, if specified with a CNP.

7.2.6 M_ASYNCH - Set Asynchronous Task Interrupt

The M_ASYNCH service resets the asynchronous task interrupt mode back to the default environment.

The nonbase mode equivalent service is M.ASYNCH.

Entry Conditions

Calling Sequence

M_ASYNCH

(or)

SVC 1,X'1C' (or) M_CALL H.REXS,68

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 asynchronous task interrupt already set

7.2.7 M_AWAITACTION - End Action Wait

The M_AWAITACTION service waits for the completion of no-wait request or I/O end action. If no such requests or end actions are outstanding, the service returns immediately. This service is similar to the M_ANYWAIT service.

The nonbase mode equivalent service is M.EAWAIT.

Entry Conditions

Calling Sequence

M_AWAITACTION [[WAITTIME=]*addr*]

(or)

```
LW   R6, addr  (or)  ZR       R6
SVC  1,X'1D'   (or)  M_CALL   H.EXEC,40
```

addr is the address containing the negative number of time units to elapse before the wait is terminated. If not specified, the task waits for an indefinite period.

Registers

R6 contains *addr*; otherwise, zero

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

EX03 USER ATTEMPTED TO GO TO AN ANY-WAIT STATE FROM AN END-ACTION ROUTINE

7.2.8 M_BACKSPACE - Backspace Record or File

The M_BACKSPACE service performs the following functions for backspacing records:

- if the file is actively generating output, issues a purge before backspacing. After the specified number of records are backspaced, returns control to the user.
- backspaces the specified number of records

M_BACKSPACE performs the following functions for blocked files:

- if the file is actively generating output, issues an end-of-file and purge backspacing, then backspaces records until an end-of-file record is found
- backspaces the specified number of files
- the read/write control word then points to the end-of-file just encountered

The nonbase mode equivalent service is M.BACK.

The M_BACKSPACE service may not be used for SYC files or unblocked files.

Entry Conditions

Calling Sequence

M_BACKSPACE [FCBADDR=]*fcbaddr*,[[MODE=] {R|F}], [NUMBER=]*number*

(or)

```
LA      R1, fcbaddr
LNW     R4, number
SVC     1,X'35'  (or)  M_CALL H.IOCS,9   } (or)
SVC     1,X'36  (or)  M_CALL H.IOCS,19   }
BIB     R4,$-W
```

fcbaddr	is the FCB address
R	specifies backspaces by record (SVC 1,X'35')
F	specifies backspaces by file (SVC 1,X'36'); this is the default
number	is the address of the word containing four times the number of records or files to backspace, or -4 if not supplied
$-W	branches back to SVC the number of times specified by *number*

Registers

R1	contains *fcbaddr*
R4	contains *number*, negated, or -4 if not supplied

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

I006	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
I009	ILLEGAL OPERATION ON THE SYC FILE

7.2.9 M_BATCH - Batch Job Entry

The M_BATCH service submits a batch job stream located in a disk file. The disk file is described by the calling parameter in R1. Prior to calling this service, the specified disk file should be rewound to purge the contents of the blocking buffer if it has been dynamically built.

The nonbase mode equivalent service is M.BATCH.

Entry Conditions

Calling Sequence

M_BATCH [ARGADR=]*arga* [,[CNP=]*cnpaddr*]

(or)

```
LW   R1,arga
LA   R7,cnpaddr  (or)  ZR R7
SVC  2,X'55'  (or)  M_CALL H.REXS,27
```

arga	is a PN vector, PNB vector, or RID vector for a permanent file; or an LFC or FCB address for a temporary file
cnpaddr	is a CNP address or zero if CNP is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN R7 (CC1 set)

(or)

M.RTRN R7 (CC1 set)

Registers

R7	return status if a CNP is not supplied; otherwise, denial address

Status

CC1 set

Posted in R7 or in the status field of the CNP:

Value	Description
0	operation successful
1	pathname invalid
2	pathname consists of volume only
3	volume not mounted
4	directory does not exist
5	disk file has not been previously opened
6	unable to activate J.SSIN2, batch job not submitted
7	resource does not exist
14	unrecoverable I/O error while reading resource descriptor
18	unrecoverable I/O error while reading directory

7.2.10 M_BBTIM - Acquire Current Date/Time in Byte Binary Format

The M_BBTIM service acquires the system date and time in byte binary format. The date and time are returned in a two word buffer, the address of which is contained in the call. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.BBTIM.

Entry Conditions

Calling Sequence

M_BBTIM [TIMBUF=] *addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'02000000'
SVC     2,X'50'  (or)  M_CALL   H.REXS,74
```

addr is the address of a 2-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

```
RX13    FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS
        OUT OF RANGE
```

7.2.11 M_BORT - Abort Specified Task, Abort Self, or Abort with Extended Message

7.2.11.1 M_BORT - Abort Specified Task

This service allows the caller to abort another task. If the named task has been swapped out, it is not aborted until it regains CPU control. See Chapter 2, Table 2-2. If the specified task is not in execution, the request is ignored.

The nonbase mode equivalent service is M.BORT.

Entry Conditions

Calling Sequence

M_BORT [ABCODE=]*abcode*, [TASK=]*task*

(or)

```
LW   R5,abcode
ZR   R6          } (or) LD R6,taskname
LW   R7,taskno   }
SVC  1,X'56'     (or) M_CALL  H.REXS,19
```

abcode contains the abort code consisting of four ASCII characters

task the address of a doubleword containing the name of the *task* or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if any of the following conditions exist:

- the specified task was not found
- the specified task name was not single copied
- the owner name of the task requesting the abort is not privileged and is restricted from access to tasks with a different owner name by the M.KEY file
- the task is in the process of exiting the system

Otherwise, contains the task number.

7.2.11.2 M_BORT - Abort Self

This service aborts the calling task by issuing an abort message, optionally performing a post-abort dump, and performing the functions common to the normal termination service as described in Chapter 2.

Entry Conditions

Calling Sequence

M_BORT [ABCODE=]*abcode*

(or)

LW R5,*abcode*
SVC 1,X'57' (or) M_CALL H.REXS,20

abcode contains the 4-character ASCII abort code

Exit Conditions

Return Sequence

M.RTRN

Output Messages

name number ABORTED AT: *xxxxxxxx-yyyyy mm/dd/yy hh:mm:ss zzzz*

name	is the 1- to 8-character name of the task being aborted
number	is the task number of the task being aborted
xxxxxxxx	is the location where the abort occurred
yyyyy	is the beginning of the DSECT
mm	is the month (2-character decimal number from 01 thru 12)
dd	is the day (2-character decimal number from 01 thru 31)
yy	is the year (2-character decimal number from 00 thru 99)
hh	is the hour (2-character decimal number from 00 thru 23)
mn	is the minutes (2-character decimal number from 00 thru 59)
ss	is the seconds (2-character decimal number from 00 thru 59)
zzzz	is the 4-character abort code

7.2.11.3 M_BORT - Abort with Extended Message

A call to this service results in an abort of the specified task with an extended abort code message.

Entry Conditions

Calling Sequence

M_BORT [ABCODE=]*abcode*, [TASK=]*task*, [EXTCODE=]*extcode*

(or)

```
LD   R2,extcode
LW   R5,abcode
LI   R6,0          } (or) LD R6,taskname
LW   R7,taskno     }
SVC  1,X'62' (or) M_CALL  H.REXS,28
```

abcode contains the abort code consisting of 4 ASCII characters

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

extcode contains the extended abort code message consisting of 1 to 8 ASCII characters, left-justified, and blank-filled

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if any of the following conditions exist:

- the specified task was not found
- the specified task name was not single copied
- the owner name of the task requesting the abort is not privileged and is restricted from access to tasks with a different owner name by the M.KEY file
- the task is in the process of exiting the system

Otherwise, contains the task number.

7.2.12 M_BRK - Break/Task Interrupt Link/Unlink

The M_BRK service allows the caller to clear the user's break receiver or to establish the address of a routine to be entered whenever another task or the operator activates his task interrupt by an M_INT service.

The nonbase mode equivalent service is M.BRK.

Entry Conditions

Calling Sequence

M_BRK [BREAKADR=]*brkaddr*

(or)

```
LA   R7,brkaddr
SVC  1,X'6E'  (or)  M_CALL   H.REXS,46
```

brkaddr is the logical word address of the entry point of the task's break/task interrupt routine or zero to clear the break receiver

Exit Conditions

Return Sequence

M.RTRN

7.2.13 M_BRKXIT - Exit from Task Interrupt Level

The M_BRKXIT service must be called at the conclusion of executing a task interrupt routine. It transfers control back to the point of interruption.

The nonbase mode equivalent service is M.BRKXIT.

Entry Conditions

Calling Sequence

M_BRKXIT

(or)

RETURN

Exit Conditions

Return Sequence

M.RTRN

7.2.14 M_BTIM - Acquire Current Date/Time in Binary Format

The M_BTIM service acquires the system date and time in binary format. The date and time are returned in a two word buffer, the address of which is contained in the call. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.BTIM.

Entry Conditions

Calling Sequence

M_BTIM [TIMBUF=]*addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'01000000'
SVC     2,X'50'  (or)  M_CALL   H.REXS,74
```

addr is the address of a 2-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.15 M_CHANPROGFCB - Execute Channel Program File Control Block

The M_CHANPROGFCB service defines a file control block (FCB) for use with an execute channel program request.

Entry Conditions

Calling Sequence

M_CHANPROGFCB [LABEL=]*label*, [LFC=]*lfc*, [CPADDR=]*addr1*,
[[TOUT=]*sec*], [[PCP=] {Y|N}], [[NWI=] {Y|N}], [[NST=] {Y|N}],
[[SENSESIZE=]*size1*], [[SENSEBUFFER=]*addr2*], [[NOWAIT=]*addr3*],
[[NOWAITERROR=] *addr4*], [[WAITERROR=] *addr5*],
[[PPCISIZE=] *size2*], [[PPCIADDR=] *addr6*]

label	is the ASCII string to use as the symbolic label for the address of this FCB
lfc	is the logical file code; word 0, bits 8-31 of the FCB
addr1	is the logical address of the channel program to be executed
sec	is the time-out value in seconds
PCP	specifies physical channel program
NWI	specifies no-wait I/O request
NST	specifies status checking not requested
size1	is the size of the user specified sense buffer
addr2	is the address of the user specified sense buffer
addr3	is the normal no-wait end-action return address
addr4	is the no-wait end-action error return address
addr5	is the wait end-action error return address
size2	is the size of PPCI status buffer to use
addr6	is the PPCI end action address

7.2.16 M_CLOSER - Close Resource

The M_CLOSER service terminates operations in the current access mode on a resource. The resource is marked closed in the file pointer table (FPT). The user count in the appropriate allocated resource table (ART) entry is decremented if implicit shared use is in effect. For access modes other than READ, the resource descriptor is updated. If last accessed functionality is enabled, the resource descriptor is also modified for read access mode. When the closing of a file implies a change of use or access mode for that resource, any tasks waiting for access to the resource in a compatible access mode are dequeued. If any logically equivalent resources are open, no further action is taken. For blocked files, any active output blocking buffer is purged. A close request to a resource that is already closed will result in an immediate return with the appropriate status posted.

The nonbase mode equivalent service is M.CLOSER.

Entry Conditions

Calling Sequence

M_CLOSER [FCBADDR=]*fcbaddr* [, [CNPADDR=]*cnpaddr*]

(or)

```
LA   R1, fcbaddr
LA   R7, cnpaddr
SVC  2,X'43'   (or)   M_CALL   H.REMM,22
```

fcbaddr is the address of a file control block (FCB)

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are abnormal return address and status field.

Registers

R1 contains *fcbaddr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA	(CC1 set)M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
8	unrecoverable I/O error to volume
29	logical file code associated with FCB does not exist
31	resource was not open
46	unable to obtain resource descriptor lock of a multiprocessor

7.2.17 M_CLSE - Close File

The M_CLSE service marks a file closed in the file pointer table (FPT) and the count of open files (DFT.OPCT) is decremented. If any logically equivalent files (ASSIGN4) are open, no further action is taken (for example, if count after decrementing is not equal to zero).

If the file is a system file or blocked file, the service purges any active output blocking buffer. The file is marked closed (open bit cleared in FAT).

For files assigned to SYC or SGO, the current disk address updates the job table for job control.

This service issues an EOF prior to purging system files SLO and SBO which were opened for read/write. It also issues an EOF before purging for blocked files that are output active.

The service ignores close requests to a file that is already closed.

The nonbase mode equivalent service is M.CLSE.

Entry Conditions

Calling Sequence

M_CLSE [FCB=]*fcbaddr* [, [EOFF=] EOF] [, [REWF=] REW]

(or)

```
LA     R1,fcbaddr
[SVC   1,X'38'  (or)  M.CALL H.IOCS, 5]
[SVC   1,X'37'  (or)  M.CALL H.IOCS, 2]
SVC    1,X'39'  (or)  M.CALL H.IOCS,23
```

fcbaddr	is the FCB address
EOF	writes EOF (SVC 1,X'38'). See the M.WEOF service description.
REW	rewinds file or device (SVC 1,X'37'). See the M.RWND service description.

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09    ILLEGAL OPERATION ON THE SYC FILE
IO38    WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE.
        A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF
        TASK HAS ONLY READ ACCESS TO UNIT.
```

7.2.18 M_CMD - Get Command Line

The M_CMD service returns the portion of the command line between the program name and the end of the line if the program name is specified on the command line. If data does not exist or the command line has already been issued, a null string is returned.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CMD.

Entry Conditions

Calling Sequence

M_CMD

(or)

SVC 2,X'61' (or) M_CALL H.REXS,88

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6	contains the length of the string in bytes, if found; otherwise, zero
R7	contains the first byte address of the string, if found; otherwise, zero

7.2.19 M_CONABB - Convert ASCII Date/Time to Byte Binary Format

The M_CONABB service converts the system date and time from ASCII format to byte binary format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONABB.

Entry Conditions

Calling Sequence

M_CONABB [ASCBUF=]*ascbuf*, [BBBUF=]*bbbuf*

(or)

```
LA      R1,ascbuf
ORMW    R1,=X'06000000'
LA      R2,bbbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

ascbuf is the address of a 4-word buffer containing the ASCII-formatted date and time

bbbuf is the address of a 2-word buffer where the byte binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.20 M_CONADB - Convert ASCII Decimal to Binary

The M_CONADB service converts ASCII decimal doublewords to their binary equivalent.

An all blank doubleword converts to zero.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONADB.

Entry Conditions

Calling Sequence

M_CONADB [[VALUEADDR=]*addr*]

(or)

```
LD   R6,addr
SVC  1,X'28' (or) M_CALL H.TSM,7
```

addr is the address of a left-justified, doubleword-bounded, ASCII-coded decimal number, blank-filled. If not specified, contents of R6 and R7 are converted.

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	contains zero if a character is nonnumeric
R7	contains the binary equivalent of the input

7.2.21 M_CONAHB - Convert ASCII Hexadecimal to Binary

The M_CONAHB service converts ASCII hexadecimal doublewords to their binary equivalent.

An all blank doubleword converts to zero.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONAHB.

Entry Conditions

Calling Sequence

M_CONAHB [[VALUEADDR=]*addr*]

(or)

```
LD   R6,addr
SVC  1,X'29'  (or)  M_CALL  H.TSM,8
```

addr is the address of a left-justified, doubleword-bounded, ASCII-coded hexadecimal number, blank-filled. If not specified, contents of R6 and R7 are converted.

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	contains zero if a character is not hexadecimal
R7	contains the binary equivalent of the input

7.2.22 M_CONASB - Convert ASCII Date/Time to Standard Binary

The M_CONASB service converts the system date and time from ASCII format to binary format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONASB.

Entry Conditions

Calling Sequence

M_CONASB [ASCBUF=]*ascbuf*, [BINBUF=]*binbuf*

(or)

```
LA      R1,ascbuf
ORMW    R1,=X'05000000'
LA      R2,binbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

ascbuf is the address of a 4-word buffer containing the ASCII formatted date and time

binbuf is the address of a 2-word buffer where the binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.23 M_CONBAD - Convert Binary to ASCII Decimal

The M_CONBAD service converts binary words to their ASCII decimal equivalent.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBAD.

Entry Conditions

Calling Sequence

M_CONBAD [[VALUEADDR=]*addr*]

(or)

```
LW   R5,addr
SVC  1,X'2A'  (or)  M_CALL   H.TSM,9
```

addr the address of a positive binary number

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6,R7 ASCII result, right-justified with leading ASCII zeros

7.2.24 M_CONBAF - Convert Binary Date/Time to ASCII Format

The M_CONBAF service converts the system date and time from binary format to ASCII format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBAF.

Entry Conditions

Calling Sequence

M_CONBAF [BINBUF=]*binbuf*, [ASCBUF=]*ascbuf*

(or)

```
LA      R1,binbuf
ORMW    R1,=X'02000000'
LA      R2,ascbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

binbuf is the address of a 2-word buffer containing the binary formatted date and time

ascbuf is the address of a 4-word buffer where the ASCII formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.25 M_CONBAH - Convert Binary to ASCII Hexadecimal

The M_CONBAH service converts binary words to their ASCII hexadecimal equivalent.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBAH.

Entry Conditions

Calling Sequence

M_CONBAH [[VALUEADDR=]*addr*]

(or)

```
LW   R5,addr
SVC  1,X'2B'  (or)  M_CALL  H.TSM,10
```

addr is the address of a binary number

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6,R7 ASCII result, right-justified with leading ASCII zeros

7.2.26 M_CONBBA - Convert Byte Binary Date/Time to ASCII

The M_CONBBA service converts the system date and time from byte binary format to ASCII format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBBA.

Entry Conditions

Calling Sequence

M_CONBBA [BBBUF=] *bbbuf*, [ASCBUF=] *ascbuf*

(or)

```
LA      R1,bbbuf
ORMW    R1,=X'04000000'
LA      R2,ascbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

bbbuf is the address of a 2-word buffer containing the byte binary formatted date and time

ascbuf is the address of a 4-word buffer where the ASCII formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.27 M_CONBBY - Convert Binary Date/Time to Byte Binary

The M_CONBBY service converts the system date and time from binary format to byte binary format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBBY.

Entry Conditions

Calling Sequence

M_CONBBY [BINBUF=] *binbuf*, [BBBUF=] *bbbuf*

(or)

```
LA      R1,binbuf
ORMW    R1,=X'01000000'
LA      R2,bbbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

binbuf is the address of a 2-word buffer containing the binary formatted date and time

bbbuf is the address of a 2-word buffer where the byte binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.28 M_CONBYB - Convert Byte Binary Date/Time to Binary

The M_CONBYB service converts the system date and time from the byte binary format to binary format. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CONBYB.

Entry Conditions

Calling Sequence

M_CONBYB [BBBUF=]*bbbuf*, [BINBUF=]*binbuf*

(or)

```
LA      R1,bbbuf
ORMW    R1,=X'03000000'
LA      R2,binbuf
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

bbbuf is the address of a 2-word buffer containing the byte binary formatted date and time

binbuf is the address of a 2-word buffer where the binary formatted date and time is returned

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.29 M_CONN - Connect Task to Interrupt

The M_CONN service indirectly connects a task to an interrupt level so that when the interrupt occurs, the specified task will be scheduled for execution (resumed). If the specified task is not active, M_CONN will preactivate it. If preactivation is required, but the actual interrupt connection is denied, M_CONN deletes the residual task because the task would continue in the suspended state indefinitely.

The nonbase mode equivalent service is M.CONN.

Entry Conditions

Calling Sequence

M_CONN [TASK=]*task*, [INTLEVEL=]*intlevel*

(or)

```
LW   R5,intlevel
LI   R6,0          } (or) LD R6,taskname (or) LW  R6, PNV
LW   R7,taskno     }                          LI  R7, 0
SVC  1,X'4B'  (or)  M.CALL   H.REXS,10
```

task the address of a doubleword containing the left-justifed blank-filled 1- to 8-character ASCII name of the task (system file only); or zero in word 0 and the task number in word 1; or pathname or RID vector in word zero and zero in word one. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

intlevel is the hardware priority level where the task is to be connected

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 Denial Code:

Value	Description
1	task already connected to an interrupt
2	another task connected to the specified interrupt
3	interrupt not SYSGEN specified indirectly connectable
4	specified task not found in dispatch queue or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file.

R7 zero if task not connected to interrupt; otherwise, contains the task number

7.2.30 M_CONSTRUCTPATH - Reconstruct Pathname

The M_CONSTRUCTPATH service constructs and returns the pathname string that was used to assign a file. In most cases, this service acquires the complete name of a file that was statically assigned to a task. If a pathname component contains special characters, the component is returned enclosed within single quotes.

The nonbase mode equivalent service is M.PNAM.

Entry Conditions

Calling Sequence

M_CONSTRUCTPATH [RESOURCE=]*addr1* , [PATH=]*addr2*
[,[CNPADDR=]*cnpaddr*]

(or)

```
LW    R1, addr1
LW    R4, addr2
LA    R7, cnpaddr    (or)  ZR      R7
SVC   2,X'2F'        (or)  M_CALL  H.VOMM,16
```

addr1	is an FCB address or LFC for the assigned volume resource
addr2	is the PN address and maximum length
cnpaddr	is a CNP address or zero if CNP is not supplied

Registers

R1	contains *addr1*
R4	contains *addr2*
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R4	M.RTRN R4
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R4	contains the actual PN length and PN address
R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.31 M_CONVERTTIME - Convert Time

The M_CONVERTTIME service returns the current date and time to the caller in the requested format. The caller must specify both the input and output format. Refer to Appendix H for date and time formats.

This service can be executed by the IPU.

Entry Conditions

Calling Sequence

M_CONVERTTIME [INBUFFER=]*inbuffer*, [INFORMAT=]*valuein* ,
[OUTBUFFER=]*outbuffer*, [OUTFORMAT=]*valueout*

(or)

SVC 2,X'51' (or) M_CALL H.REXS,75

inbuffer specifies the address of the input buffer

valuein is the keyword for the input format. The three valid keywords are:

BIN 2-word internal binary value:
word 1 is the number of days since January 1, 1960
word 2 is the number of clock ticks since midnight

BYTE 8-byte binary value:

Byte	Contents in Binary
0	century
1	year
2	month
3	day
4	hour
5	minute
6	second
7	number of interrupts

QUAD 4-word ASCII string formatted the same as BYTE

outbuffer specifies the address of the output buffer

valueout is the keyword for the output format. The three valid keywords are the same as for *valuein*.

Registers

R1 contains *inbuffer*

R2 contains *outbuffer*

Exit Conditions

Return Sequence

M.RTRN

Registers

R1,R2 used by the call

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.32 M_CREATEFCB - Create File Control Block

The M_CREATEFCB service defines an expanded file control block (FCB).

Entry Conditions

Calling Sequence

M_CREATEFCB [LABEL=] *label* ,[LFC=] *lfc* , [COUNT=] *count* ,
[DATABUFFER=] *addr1*, [[WAITERROR=]*addr2*], [[RANDOM=]*addr3*],
[[NWT=] {Y|N}], [[NER=] {Y|N}], [[DFI=] {Y|N}], [[NST=] {Y|N}],
[[RAN=] {Y|N}], [[BIN=] {Y|N}], [[SKIPLDR=] {Y|N}],
[[PACKED=] {Y|N}], [[PARITY=] {O|E}], [[BPI=] {556|800}],
[[NOWAIT=]*addr4*], [[NOWAITERROR=]*addr5*], [[BLOCKBUFFER=]*addr6*]

label	is an ASCII string to be used as symbolic label for the address of this FCB
lfc	is the logical file code
count	is the transfer count specified in number of bytes
addr1	is the start address of a data buffer, reserved on a word boundary
addr2	is the error return address for wait I/O
addr3	is the random access address
NWT	specifies the control flag for no-wait I/O
NER	specifies the control flag for error return processing
DFI	specifies the control flag for data format
NST	specifies the control flag for status checking by the handler
RAN	specifies the control flag for random access
BIN	specifies the control flag for use with a card reader
SKIPLDR	specifies the control flag for use with a paper tape reader
PACKED	specifies the control flag for use with 5-track magnetic tape
PARITY	indicates even or odd parity
BPI	designates bits per inch density
addr4	specifies the address for normal no-wait I/O end action return
addr5	specifies the address for no-wait I/O error end action return
addr6	specifies the address of a 192-word user-supplied blocking buffer to be used for blocked I/O

7.2.33 M_CREATEP - Create Permanent File

The M_CREATEP service creates a permanent file. Permanent files are given names in directories and remain known to the operating system until explicitly deleted.

This service allocates a resource descriptor and the initial file space requirements for the file. Next, the specified attributes of the file are recorded in the resource descriptor. As a final step, the name of the file is established in the indicated directory.

When a directory entry is established, the directory entry is linked to the resource descriptor of the file. This links the name of the file to the other attributes of the file. Typical file attributes are:

- file name
- file resource identifier (RID)
- file protection attributes
- file management attributes
- file initial space requirements

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.CPERM.

Entry Conditions

Calling Sequence

M_CREATEP [PNADDR=]*addr* [,[RCBADDR=]*rcbaddr*] [,[CNPADDR=]*cnpaddr*]

(or)

```
LW   R1, addr
LA   R2, rcbaddr (or) ZR     R2
LA   R7, cnpaddr (or) ZR     R7
SVC  2,X'20'(or) M_CALL  H.VOMM,1
```

addr	is an address containing a PN vector or PNB vector
rcbaddr	is an RCB address or zero if default attributes are desired
cnpaddr	is a CNP address or zero if CNP is not supplied

Registers

R1	contains *addr*
R2	contains *rcbaddr*; otherwise, zero
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.34 M_CREATET - Create Temporary File

The M_CREATET service creates a temporary file. Temporary files are not given names in directories and remain known to the operating system only for as long as the task that created them is in execution. Typically, when a task terminates execution (normally or abnormally), its temporary files are deleted by MPX-32.

If the temporary file is made permanent or is allocated (assigned) to another task when the creator terminates execution, the file remains defined to MPX-32 after the task that created it terminates execution.

This service allocates a resource descriptor for the file and acquires the initial space requirements for the file. As a final step, it records the attributes of the file in the resource descriptor.

When a temporary file is created, the typical file attributes are:

- file resource identifier (RID)
- file protection attributes
- file management attributes
- file initial space requirements

The file's RID is returned only if the RCB address is specified and an ID location address for the file is also specified within the RCB.

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.TEMP.

Entry Conditions

Calling Sequence

M_CREATET [[PATH=] *vector*] [,[RCBADDR=]*rcbaddr*] [,[CNPADDR=]*cnpaddr*]

(or)

```
LW    R1 vector      (or)   ZR      R1
LA    R2, rcbaddr    (or)   ZR      R2
LA    R7, cnpaddr    (or)   ZR      R7
SVC   2,X'21'        (or)   M_CALL  H.VOMM,2
```

vector is an address containing a PN (volume name only) vector or zero if the file to be created is on a working volume

rcbaddr is a RCB address or zero if default attributes are desired

cnpaddr is a CNP address or zero if CNP is not supplied

Registers

R1	contains *vector*; otherwise, zero
R2	contains *rcbaddr*; otherwise, zero
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.35 M_CTIM - Convert System Date/Time Format

The M_CTIM service converts the system date and time from one of three standard formats (see Appendix H for date and time formats) to either of the other two formats. This service is callable from specific case macros that provide the function code in the macro call itself.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.CTIM.

Entry Conditions

Calling Sequence

M_CTIM [FUNCT=]*funct*, [STIME=]*addr1* , [DTIME=]*addr2*

(or)

```
LA      R1,addr1
ORMW    R1,funct
LA      R2,addr2
SVC     2,X'51'  (or)  M_CALL   H.REXS,75
```

funct is the address of a word that contains the function code (see chart below) in byte 0 (most significant) and zeros in bytes 1, 2, and 3

addr1 is the address of a 2- or 4-word buffer where the user provides the date and time, in any of the three standard formats, for the system to convert

addr2 is the address of a 2- or 4-word buffer where the system returns the converted date and time values in the format requested by the user

Function Code	Input Format	Return Format	Buffer Length In	Buffer Length Out
1	Binary	Byte binary	2W	2W
2	Binary	Quad ASCII	2W	4W
3	Byte binary	Binary	2W	2W
4	Byte binary	Quad ASCII	2W	4W
5	Quad ASCII	Binary	4W	2W
6	Quad ASCII	Byte binary	4W	2W

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1,R2 used by call; all others returned intact

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.36 M_CWAT - System Console Wait

The M_CWAT service suspends operation of the calling program until the specified I/O transfer is complete.

The nonbase mode equivalent service is M.CWAT.

Entry Conditions

Calling Sequence

M_CWAT [TCP=]*tcpb*

(or)

```
LA   R1,tcpb
SVC  1,X'3D'  (or)  M_CALL  H.IOCS,26
```

tcpb is the address of a type control parameter block (TCPB)

Exit Conditions

Return Sequence

M.RTRN

7.2.37 M_DATE - Date and Time Inquiry

The M_DATE service returns to the caller the date (in ASCII), calendar information (century, year, month and day), and a count of the number of real-time clock interrupts since midnight. To aid in converting the interrupt count to time-of-day, the service also returns counts of the number of interrupts per second and the number of interrupts per time unit.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.DATE.

Entry Conditions

Calling Sequence

M_DATE [PBADR=]*pbaddr*

(or)

```
LA   R7,pbaddr
SVC  1,X'15'  (or)  M_CALL  H.REXS,70
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

- Words 0 and 1 contain the current date in the format entered at IPL time.
- Word 2

Bytes	Contents in Binary
0	century
1	year
2	month
3	day

- Word 3 contains the number of clock interrupts since midnight
- Word 4 contains the number of clock interrupts per second (initialized by SYSGEN)
- Word 5 contains the number of clock interrupts per time unit (initialized by SYSGEN)

Exit Conditions

Return Sequence

M.IPURTN

7.2.38 M_DEASSIGN - Deassign and Deallocate Resource

The M_DEASSIGN service deallocates a resource and disassociates it from a logical file code. When a device associated with any unformatted media is detached, a message is issued to inform the operator to dismount the medium, unless inhibited by user request or system constraints.

Deallocation of a nonshared resource makes it available to other tasks. Deallocation of a shared resource makes the resource available, if the caller is the last task to deallocate it or the access mode changes as a result of the deallocation to allow other compatible tasks to attach to the resource. Deallocation of SLO and SBO files result in their definitions being passed to the system output task for processing.

If the specified logical file code has been equated to other logical file codes in the system, only the specified LFC is deallocated. If a close has not been issued, the resource is also closed. This function can also issue a dismount message for an unformatted medium with no resource deallocation.

The nonbase mode equivalent service is M.DASN.

Entry Conditions

Calling Sequence

M_DEASSIGN [RESOURCE=]*addr* [,[CNPADDR=]*cnpaddr*]

(or)

```
LW   R1, addr
LA   R7, cnpaddr (or) ZR      R7
SVC  2,X'53'  (or) M_CALL  H.REXS,22
```

addr is an address containing the allocation index obtained when the resource was assigned

(or)

an address containing the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are abnormal return address, option field, and status field.

The option field of the CNP has the following bit significance when set:

0 - issue dismount message with no resource deallocation

Registers

R1 contains *addr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or in the status field of the CNP:

Value	Description
8	unrecoverable I/O error to volume
29	logical file code associated with FCB not assigned
30	invalid allocation index
46	unable to obtain resource descriptor lock (multiprocessor only)

7.2.39 M_DEBUG - Load and Execute Interactive Debugger

The M_DEBUG service causes one of the following events to occur:

- If the interactive debugger is currently loaded at the time the service is called, control is transferred to the debugger.
- If the interactive debugger is not currently loaded at the time the service is called, the debugger is loaded as an overlay segment, then control is transferred to the debugger.

DEBUGX32 is loaded with base mode tasks.

The nonbase mode equivalent service is M.DEBUG.

Entry Conditions

Calling Sequence

M_DEBUG

(or)

SVC 1,X'63' (or) M_CALL H.REXS,29

Exit Conditions

Normal Return Sequence to Debugger:

M.RTRN R7 contains the transfer address of the debugger if the debugger was loaded by this service call. R7 contains zero if the debugger was already loaded at the time this service was called.

Abnormal Return Sequence to Caller:

R7 Value	Description
2	debugger load module not found
4	invalid preamble
5	insufficient task space for loading
6	I/O error on resource descriptor
7	I/O error on resource
8	loading error

7.2.40 M_DEFT - Change Defaults

The M_DEFT service changes the caller's working directory, project group protection, or both.

The caller should invoke this service with two separate calls, as though changing project group protection and changing the current working directory were two distinct services. When both project group and working directory are specified, the service changes the project group, before attempting to change the current working directory. If the attempt to establish the current working directory fails, the new project group protection remains in effect and the caller is notified by an error status code that the current working directory request failed. The caller must determine whether to continue with the new project group or to re-establish another project group.

The nonbase mode equivalent service is M.DEFT.

Entry Conditions

Calling Sequence

M_DEFT [ARGA=]*arga* [, [CNP=]*cnpaddr*] [, [PRJADR=] *prjaddr*]
[, [KEYADR=] *keyaddr*]

(or)

```
LW   R1,arga
LA   R4,prjaddr  (or) ZR  R4
LA   R5,keyaddr  (or) ZR  R5
LA   R7,cnpaddr  (or) ZR  R7
SVC  2,X'27'  (or) M_CALL  H.VOMM,8
```

arga	contains a PN vector, PNB vector, or RID vector
cnpaddr	is a CNP address or zero if CNP is not supplied
prjaddr	is the address of the new project group name or zero if no change to project group name
keyaddr	is the address of the new project group key or zero if not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.41 M_DELETER - Delete Resource

The M_DELETER service explicitly deletes volume resources. This service must be used to delete directories, files, and memory partitions. Callers cannot delete a resource that they do not have delete access to.

This service first deletes the directory entry for the specified resource and then releases the volume space requirements. Finally the service releases the resource descriptor.

If the resource is allocated at the time of the delete request, only the directory entry is deleted. The volume space requirements and the resource descriptor for the resource will be released when the last assignment to the resource is removed.

To delete a permanent file or memory partition, the pathname or pathname block must be supplied. To delete a directory, the pathname or pathname block must be supplied and all files which were defined in the directory must have been previously deleted.

To delete a temporary file, the caller can provide the resource identifier (RID), or specify the logical file code (LFC), or the address of a file control block (FCB).

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.DELR.

Entry Conditions

Calling Sequence

M_DELETER [RESOURCE=]*addr* [,[CNPADDR=]*cnpaddr*]

(or)

```
LW   R1, addr
LA   R7, cnpaddr (or) ZR  R7
SVC  2,X'24'    (or)  M_CALL  H.VOMM,5
```

addr is an address containing a PN vector, PNB vector, LFC, or FCB

cnpaddr is a CNP address or zero if CNP is not supplied

Registers

R1 contains *addr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.42 M_DELTSK - Delete Task

The M_DELTSK service forces I/O completion and immediately aborts the specified task. See Task Termination Sequencing in Chapter 2. This service should only be used when abort fails to remove a task or when a task is queued for a resource. File integrity can be affected because operations are not allowed to complete normally. To preserve system integrity, the kill directive is processed as an abort for ten seconds. If this does not remove the task, it is killed.

The nonbase mode equivalent service is M.DELTSK.

Entry Conditions

Calling Sequence

M_DELTSK [ABCODE=]*abcode*, [TASK=]*task*, [EXTCODE=]*extcode*

(or)

```
LD   R2,extcode
LW   R5,abcode
LI   R6,0          } (or) LD R6,taskname
LW   R7, taskno    }
SVC  1,X'5A'  (or) M_CALL  H.REXS,31
```

abcode contains the abort code consisting of four ASCII characters

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

extcode contains the extended abort code message consisting of 1- to 8-ASCII characters, left-justified, and blank-filled

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, contains the task number

Output Messages

Modifies abort message to:

ABORT *task* REASON: *xxxx zzzzzzzz* AT: *yyyyyyyy*

zzzzzzzz is the extended message code supplied with the call to this service

7.2.43 M_DEVID - Get Device Mnemonic or Type Code

The M_DEVID service allows the user to pass a device mnemonic or a generic device type code and receive the corresponding type code or mnemonic. See Appendix A for device mnemonics and device type codes.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.DEVID.

Entry Conditions

Calling Sequence

M_DEVID [DEVID=] *id*

(or)

```
LW   R2,id
SVC  1,X'14' (or)  M_CALL   H.REXS,71
```

id contains either a device mnemonic in the right halfword with the left halfword zero or a device type code in byte 3 with zero in bytes 0-2

Exit Conditions

Return Sequence

M.IPURTN 2

Registers

Registers if input was a device mnemonic:

R2 bytes 0-2 contain zeros; byte 3 contains corresponding device type code

R2 left halfword contains zero; right halfword contains corresponding device mnemonic

Registers if input was a mnemonic or device type code not in the system device type table (DTT):

R2 bit 0 is set; bits 1-31 are unchanged

7.2.44 M_DIR - Create Directory

The M_DIR service creates a permanent directory. Permanent directories are given names in the root directory and remain known to MPX-32 until they are explicitly deleted.

Directories contain the names of permanent files and memory partitions that are created in the directories.

This service allocates a resource descriptor and the volume space requirements for the directory. Next, it records the indicated attributes of the directory in the resource descriptor. Finally, the service establishes the name of the directory in the indicated parent directory.

When the directory is established, the directory entry is linked to the resource descriptor of the directory. This links the name of the new directory to the other attributes of the new directory. Typical directory attributes are:

- directory name
- directory resource identifier (RID)
- directory protection attributes
- directory management attributes
- directory volume space requirements

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.DIR.

Entry Conditions

Calling Sequence

M_DIR [PNADR=]*pnaddr* [,[CNP=]*cnpaddr*] [,[RCB=]*rcbaddr*]

(or)

```
LW   R1,pnaddr
LA   R2,rcbaddr  (or) ZR  R2
LA   R7,cnpaddr  (or) ZR  R7
SVC  2,X'23'  (or) M_CALL  H.VOMM,4
```

pnaddr	contains a PN vector or PNB vector
cnpaddr	is a CNP address or zero if CNP not supplied
rcbaddr	is a RCB address or zero if default attributes are desired

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.45 M_DISCON - Disconnect Task from Interrupt

The M_DISCON service disconnects a task that has previously been connected to an interrupt level.

The nonbase mode equivalent service is M.DISCON.

Entry Conditions

Calling Sequence

M_DISCON [TASK=]*task*

(or)

```
LI    R6,0          } (or) LD R6,taskname
LW    R7,taskno     }
SVC   1,X'5D'       (or) M_CALL  H.REXS,38
```

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Normal Return Sequence

M.RTRN 7

Registers

R7 contains the task number

Abnormal Return Sequence

M.RTRN 6,7

Registers

R6 contains denial code as follows:

Value	Description
1	task not found in dispatch queue or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file
2	task not indirectly connected
3	task connected to invalid interrupt

R7 zero if the task was not previously connected to an interrupt level

7.2.46 M_DISMOUNT - Dismount Volume

The M_DISMOUNT service performs a logical dismount and, optionally, a physical dismount of a user volume.

For a nonpublic volume, M_DISMOUNT decrements the use count for the volume in the mounted volume table (MVT) entry, provided the requestor has no resources allocated on the volume. A physical dismount is performed if the MVT use count is zero. Otherwise, the physical dismount is pending, and all mount requests for the volume are denied.

For a physical dismount of a public volume, M_DISMOUNT establishes a use count in the MVT based on the total number of resources allocated on the volume at the time the dismount request is made. When the use count is zero, a physical dismount is performed. M_DISMOUNT confirms the completed dismount with the system operator through the system console, as specified in the CNP.

Only the system administrator can request the dismount of a public volume.

The nonbase mode equivalent service is M.DMOUNT.

Entry Conditions

Calling Sequence

M_DISMOUNT [VOLADDR=]*voladdr* [,[CNPADDR=]*cnpaddr*]

(or)

```
LA   R1, voladdr
LA   R7, cnpaddr  (or) ZR      R7
SVC  2,X'4A'  (or) M_CALL  H.REMM,19
```

voladdr is the address, doubleword bounded of the volume name

cnpaddr is the address of a caller notification packet if notification is desired

Applicable portions of the CNP for this function are option field, abnormal return address, status field, and Word 3.

Option field (Word 2 of CNP):

Bit	Meaning if Set
0	physical dismount requested
1	no logical dismount
2	public volume dismount request
3	inhibit operator interaction

Note: Bit 3 is ignored if the volume was mounted with operator intervention inhibited.

Word 3: Dismount device specification, if the option bit is set

Byte	Definition
1	device type code
2	channel address
3	subchannel address

Registers

R1	contains *addr*
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or in the status field of the CNP:

Value	Description
14	caller has outstanding resource assignments on this volume
20	volume not assigned to this task or volume is public
86	cannot dismount the system volume
87	unable to dismount public volume because compatible mode public volume dismount (CMPMM) option was specified at SYSGEN
88	unable to dismount public volume. SA attribute required.
89	unable to dismount public volume. Missing CNP option on dismount request.

7.2.47 M_DLTT - Delete Timer Entry

The M_DLTT service resets the timer for the specified task so that its specified function is no longer performed upon time-out. Deletion of the timer entry does not delete the associated task. One-shot timers are deleted on expiration.

The nonbase mode equivalent service is M.DLTT.

Entry Conditions

Calling Sequence

M_DLTT [TIMER=] *timer*

(or)

```
LW   R7,timer
SVC  1,X'47'  (or)  M_CALL   H.REXS,6
```

timer specifies right-justified 2-character ASCII name of a timer

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 *timer* entry not found

7.2.48 M_DSMI - Disable Message Task Interrupt

The M_DSMI service disables the task interrupts for messages sent to the calling task. M_DSMI is useful for synchronization gating of the task message interrupts.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.DSMI.

Entry Conditions

Calling Sequence

M_DSMI

(or)

SVC 1,X'2E' (or) M_CALL H.REXS,57

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1 task interrupts were already disabled

7.2.49 M_DSUB - Disable User Break Interrupt

The M_DSUB service deactivates the user break interrupt (see M_ENUB) and allows user breaks by the terminal break key to be acknowledged.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.DSUB.

Entry Conditions

Calling Sequence

M_DSUB

(or)

SVC 1,X'12' (or) M_CALL H.REXS,73

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1 user break already disabled

7.2.50 M_DUMP - Memory Dump Request

The M_DUMP service provides a dump of the caller's program status doubleword (PSD), general purpose registers, and specified memory limits. The output is to a SLO file in side-by-side hexadecimal with ASCII format, with the PSD and in registers preceding the specified memory limits. The PSD and registers are extracted from the first level of push-down of the calling task. Optionally, R5 can specify the address of a ten word block containing R0 through R7 and the PSD to be dumped, respectively.

Any task can request a memory dump.

The nonbase mode equivalent service is M.DUMP.

Entry Conditions

Calling Sequence

M_DUMP [START=]*start*, [ENDADR=]*end* [,[CTXBUF=]*ctxbuf*]

(or)

```
ZR   R5 (or)  LA   R5,ctxbuf
LW   R6,start
LW   R7,end
SVC  1,X'4F'  (or)  M_CALL   H.REXS,12
```

start contains the low logical word address requested in dump

end contains the high logical word address requested in dump

ctxbuf is the optional address of ten consecutive words containing R0 through R7 and a PSD, respectively. If R5=0, the registers and PSD dumped are taken from the first level of push-down.

Note: *start* and *end* are truncated to the nearest 8-word boundaries and memory is dumped between the truncated limits.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6

Value	Description
1	high dump limit less than low limit
4	no FAT or FPT space available
5	request made with insufficient levels of push-down available
6	cannot allocate SLO file
7	unrecoverable I/O error

R7 zero if dump could not be performed

7.2.51 M_ENMI - Enable Message Task Interrupt

The M_ENMI service enables task interrupts for messages sent to the calling task. It removes an inhibit condition previously established by invoking the M_DSMI service.

The nonbase mode equivalent service is M.ENMI.

Entry Conditions

Calling Sequence

M_ENMI

(or)

SVC 1,X'2F' (or) M_CALL H.REXS,58

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set task interrupts were already enabled

7.2.52 M_ENUB - Enable User Break Interrupt

The M_ENUB service activates the user break interrupt and causes further user breaks from the user terminal break key to be ignored.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.ENUB.

Entry Conditions

Calling Sequence

M_ENUB

(or)

SVC 1,X'13' (or) M_CALL H.REXS,72

Exit Conditions

Return Sequence

M.IPURTN

Status

CC1 set user break already enabled

7.2.53 M_ENVRMT - Get Task Environment

The M_ENVRMT service obtains more information on the task environment than provided in the task option word.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.ENVRMT.

Entry Conditions

Calling Sequence

M_ENVRMT

(or)

SVC 2,X'5E' (or) M_CALL H.REXS,85

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 contains the task environment word as follows:

Bit	Definition
0	0 if batch task or real-time task 1 if interactive
1	0 if option NOCOMMAND is set 1 if option COMMAND is set
2	0 if option NOERR is set 1 if option ERROR is set
3	0 if cataloged or linked unprivileged 1 if cataloged or linked privileged
4	0 if currently unprivileged 1 if currently privileged
5-6	reserved
7	0 if task is not in demand page mode 1 if task is in demand page mode (CONCEPT 32/2000 only)
8-31	reserved

7.2.54 M_EXCLUDE - Exclude Shared Image

The M_EXCLUDE service allows a base mode task to dynamically exclude a shared image previously included. This service causes the assign count and user count to be decremented. The shared image is deleted and its resources returned to the free list when the assign count goes to zero. This service is also called by the exit processor (H.REMM,3) whenever a task aborts or abnormally ends while associated with a shared image. The shared image is identified by the allocation index obtained when the shared image was included, or by the resource identifier (RID) used to create it and the owner name used to include it.

The nonbase mode equivalent service is M.EXCLUDE.

Entry Conditions

Calling Sequence

M_EXCLUDE [RESOURCE=] *addr* [,[CNPADDR=]*cnpaddr*]

(or)

```
LW   R1, addr
LA   R7, cnpaddr (or) ZR  R7
SVC  2,X'41' (or) M_CALL H.REMM,14
```

addr contains a PN vector, or PNB vector, an RID vector, or the allocation index obtained from the M_INCLUDE service

cnpaddr is a CNP address or zero if CNP is not supplied. Applicable portions of the CNP are abnormal return address and status field.

Registers

R1 contains *addr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R3, R5	M.RTRN R3, R5
(or)	(or)
M.RTRNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
5	unable to locate shared memory table entry
30	invalid allocation index
35	attempt to exclude shared image that is not mapped into the calling task's address space
39	unable to write back data section
58	shared memory table space is not available

7.2.55 M_EXIT - Terminate Task Execution

The M_EXIT service performs all normal termination functions required of exiting tasks. All devices and memory are deallocated, related table space is erased, and the task's dispatch queue entry is cleared.

The nonbase mode equivalent service is M.EXIT.

Entry Conditions

Calling Sequence

M_EXIT

(or)

SVC 1,X'55' (or) M_CALL H.REXS,18

R0 must be preloaded with an ASCII abort code or zero. R0 must contain zero for a normal exit.

Exit Conditions

Return Sequence

M.RTRN R0

Registers

R0 contains abort code on abnormal exits.

Abort Cases

RX92 TASK HAS ATTEMPTED NORMAL EXIT WITH MESSAGES IN ITS RECEIVER QUEUE

7.2.56 M_EXTENDFILE - Extend File

The M_EXTENDFILE service allows the space of a file to be manually extended. The caller may specify the size of the requested extension or use the default file extension parameters defined when the file was created. If the file was created with the zero option specified, the extension will be zeroed.

This service will only extend temporary or permanent files that are manually extendable. Directories and memory partitions cannot be extended. The caller must have write, update, or append access in order to extend the file.

The caller can extend a file regardless of whether the file is currently allocated. Additionally, the caller can supply any allowable resource specification, such as pathname (PN), pathname block (PNB), resource ID (RID), logical file code (LFC) or address of a file control block (FCB).

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.EXTD.

Entry Conditions

Calling Sequence

M_EXTENDFILE [RESOURCE=] *addr* [,[BLOCKS=] *number*]
[,[CNPADDR=] *cnpaddr*]

(or)

```
LW   R1, addr
LN   R6, number  (or)  ZR  R6
LA   R7, cnpaddr (or)  ZR R7
SVC  2,X'25'  (or) M_CALL  H.VOMM,6
```

addr	contains a PN vector, a PNB vector, an RID vector, an LFC, or an FCB
number	is an address containing the number of blocks to extend the file by or zero if RCB extension parameters specified during file creation are to be used
cnpaddr	is a CNP address or zero if not supplied

Registers

R1	contains *addr*
R6	contains *number*; otherwise, zero
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R6	M.RTRN R6
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R6	contains the number of contiguous blocks the file is actually extended by
R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.57 M_EXTSTS - Exit With Status

The M_EXTSTS service provides exit status of a task, i.e., whether the exit was successful. R0 contains either zero or a valid 4-character ASCII abort code. If R0 is not zero or a valid abort code, the task is terminated with an RX36 abort condition.

Entry Conditions

Calling Sequence

M_EXTSTS [STATADDR=] *addr*

(or)

LW R0, *addr*
SVC 2,X'5F' (or) M_CALL H.REXS,86

addr is the address where the final status code is to be stored

Registers

R0 contains *addr*

Exit Conditions

Abort Cases

RX36 STATUS IN R0 IS NOT A ZERO OR A VALID ABORT CODE

Output Messages

task #*number* ABORT AT: *xxxxxxxx - yyyyy mm/dd/yy hh:mn:ss zzzz*

task	is the 1- to 8-character name of the task being aborted
number	is the task number of the task being aborted
xxxxxxxx	is the location of the abort
yyyyy	is the beginning of the DSECT
mm	is the month (2-character decimal number from 01 thru 12)
dd	is the day (2-character decimal number from 01 thru 31)
yy	is the year (2-character decimal number from 00 thru 99)
hh	is the hour (2-character decimal number from 00 thru 23)
mn	is the minutes (2-character decimal number from 00 thru 59)
ss	is the seconds (2-character decimal number from 00 thru 59)
zzzz	is the 4-character abort code

7.2.58 M_FREEMEMBYTES - Free Memory in Byte Increments

The M_FREEMEMBYTES service allows a task to dynamically deallocate acquired memory. Deallocation can be random. The space address must have been previously obtained from the M_GETMEMBYTES service. All of the space obtained from a given call is deallocated.

The nonbase mode equivalent service is M.MEMFRE.

Entry Conditions

Calling Sequence

M_FREEMEMBYTES [MEMADDR=]*addr*

(or)

```
LW   R3, addr
SVC  2,X'4C'   (or)   M_CALL   H.REMM,29
```

addr is the starting address of dynamic space previously acquired from the M_GETMEMBYTES service

Registers

R3 contains *addr*

Exit Conditions

Return Sequence

M.RTRN R3 (or) abort user with RM77

Registers

R3 equals zero if deallocation could not be performed. Deallocation address was not found in allocation table.

Abort Cases

RM77 A TASK HAS DESTROYED THE ALLOCATION LINKAGES IN ITS DYNAMIC EXPANSION SPACE

7.2.59 M_GETCTX - Get User Context

The M_GETCTX service is used to store the current values for user context into a specified buffer. The buffer must be on a word boundary.

Entry Conditions

Calling Sequence

M_GETCTX [BUFFER=] *addr* , [NUMBER=] *number*

(or)

```
LA   R1, addr
LI   R4, number
SVC  2,X'70'   (or)   M_CALL   H.EXEC,41
```

addr is a word bounded buffer address where the context is to be stored

number is the number of bytes allocated for the buffer

Registers

R1 contains *addr*

R4 contains *number*

Exit Conditions

Return Sequence

M.RTRN R6, R7

Registers

Normal Return:

CC1 = 0

R6 contains debugger status word 1

R7 contains debugger status word 2

Return On Error Condition

CC1 = 1

R7 contains error condition as a hexadecimal value as follows:

Value	Description
257	destination buffer is in the operating system
258	destination buffer is in the TSA
259	invalid destination buffer address
260	buffer length not a word multiple

7.2.60 M_GETDEF - Get Definition for Terminal Function

The M_GETDEF service returns, for the requested terminal function, an appropriate string of bytes in the specified buffer and indicates the length of the returned string. The user must specify the LFC of a terminal that is currently open and allocated, the buffer address and a terminal function. For this service to operate, the partition TERMPART must exist and have been initialized by J.TDEFI. For more information, refer to the MPX-32 Reference Manual, Volume II, Chapter 11.

The nonbase mode equivalent service is M.GETDEF.

Entry Conditions

Calling Sequence

M_GETDEF [IBADDR=] *ibaddr*

(or)

```
LA   R1, ibaddr
SVC  2,X'7A' (or) M_CALL H.TSM,15
```

ibaddr is the logical 24-bit word address of the first word of the TERMDEF information block formatted as follows:

Word	Description
0	open and allocated terminal's LFC
1	user buffer 24 bit address for returned information
2	halfword 0 is the requested function (2 ASCII alphanumeric characters); halfword 1 is reserved
3	halfword 0 is the user buffer length in bytes; halfword 1 is the length in bytes of the string returned by the service to the user's buffer
4	optional X coordinate for cursor positioning functions
5	optional Y coordinate for cursor positioning functions

R1 is assumed to contain the address if *ibaddr* is not supplied.

Exit Conditions

Return Sequence

M.RTRN On normal completion, the string for the requested function is in the user's buffer, and the length of this string is in the TERMDEF information block string length field.

CC1 set Error detected. The string length in the TERMDEF information block is set to 0 and the function contains the error number with the following meanings:

function=error

error	Description
1	invalid LFC supplied
2	unknown terminal type
3	user buffer is too large (>2K)
4	cannot include partition
5	undefined function requested
6	user buffer is too small
7	partition data integrity suspect
8	invalid terminal type supplied
9	invalid user buffer address
10	function is invalid for this terminal
11	TERMDEF is not installed
N/A	TERMDEF information block address is invalid (CC1 set only)

Registers

R1 unchanged. CC1 set if an error is detected.

Abort Cases

MF01 A MAP FAULT TRAP HAS OCCURRED. THIS IS THE RESULT OF A BAD MEMORY REFERENCE OUTSIDE OF THE USER'S ADDRESSABLE SPACE.

7.2.61 M_GETMEMBYTES - Get Memory in Byte Increments

The M_GETMEMBYTES service allows a task to dynamically expand its memory allocation in doubleword increments starting at the end of its DSECT up to the end of its logical address space. The additional memory is of the same type specified when the task was linked. The task is mapped in a logically contiguous manner up to the end of its address space. The task is suspended until the allocation is successful. Repeated calls to this service are allowed. Allocation is not contiguous with previously allocated space.

The nonbase mode equivalent service is M.MEMB.

Entry Conditions

Calling Sequence

M_GETMEMBYTES [NUMBER=]*number*

(or)

```
LI    R4, number
SVC   2,X'4B'   (or)   M_CALL   H.REMM,28
```

number is the number of bytes to allocate

Registers

R4 contains *number*

Exit Conditions

Return Sequence

M.RTRN R3,R4

Registers

CC1	equals zero
CC2	equals zero
R3	contains the 24-bit starting logical doubleword address of allocated space
R4	contains the number of bytes actually allocated, modulo 2W

(or)

CC1	equals zero
CC2	equals one
R3	contains the 24-bit starting logical doubleword address of allocated space
R4	contains the number of bytes actually allocated (modulo 2W); however, the number is less than requested.

Error Condition

Allocation Denied:

CC1	equals one
CC2	equals one
R3	equals zero
R4	equals zero

7.2.62 M_GETTIME - Get Current Date and Time

The M_GETTIME service returns the current date and time to the caller in any one of the three standard formats described in Appendix H.

This service can be executed by the IPU.

Entry Conditions

Calling Sequence

M_GETTIME [OUTBUFFER=]*addr*,[OUTFORMAT=]*value*
(or)
LA R1,*addr*
ORMW R1,X'01000000' (or) ORMW R1,X'02000000' (or)
ORMW R1,X'03000000'
(or)
SVC 2,X'50' (or) M_CALL H.REXS,74

addr specifies the address of the output buffer

value is the keyword for the format in which the date and time will be returned. The three valid keywords are:

Function Code	Keyword	Format
1	BIN	2-word internal binary value as follows:

Word	Contents
1	number of days since January 1, 1960
2	number of clock ticks since midnight

Function Code	Keyword	Format
2	BYTE	8-byte binary value as follows:

Byte	Contents in Binary
0	century
1	year
2	month
3	day
4	hour
5	minute
6	second
7	number of interrupts

Function Code	Keyword	Format
3	QUAD	4-word ASCII string formatted as follows:

Halfword	Contents in ASCII
0	century
1	year
2	month
3	day
4	hour
5	minute
6	second
7	number of interrupts

Registers

R1 byte 0 contains the function code; bytes 1-3 contain *addr*

Exit Conditions

Return Sequence

M.RTRN

Registers

R1 byte 0 contains the function code and bytes 1-3 contain the address as used by the call. All others are returned intact.

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE

7.2.63 M_GMSGP - Get Message Parameters

The M_GMSGP service is called from the message receiver routine of a task that has received a message interrupt. It transfers the message parameters into the designated receiver buffer, and posts the owner name and task number of the sending task into the parameter receive block (PRB). For description of the PRB, see Chapter 2.

The nonbase mode equivalent service is M.GMSGP.

Entry Conditions

Calling Sequence

M_GMSGP [PRB=]*prbaddr*

(or)

```
LA   R2,prbaddr
SVC  1,X'7A'  (or)  M_CALL  H.REXS,35
```

prbaddr is the logical address of the parameter receive block (PRB)

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing status error code:

Value	Description
0	normal status
1	invalid PRB address
2	invalid receiver buffer address or size detected during parameter validation
3	no active send request
4	receiver buffer length exceeded during transfer

7.2.64 M_GRUNP - Get Run Parameters

The M_GRUNP service transfers the run parameters into the designated receiver buffer, and posts the owner name and task number of the sending task into the parameter receive block (PRB). It is called by a task that is executing for a run request. See Chapter 2.

The nonbase mode equivalent service is M.GRUNP.

Entry Conditions

Calling Sequence

M_GRUNP [PRB=]*prbaddr*

(or)

```
LA   R2,prbaddr
SVC  1,X'7B'  (or)  M_CALL  H.REXS,36
```

prbaddr is the logical address of the parameter receive block (PRB)

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing status error code:

Value	Description
0	normal status
1	invalid PRB address
2	invalid receiver buffer address or size detected during parameter validation
3	no active send request
4	receiver buffer length exceeded during transfer

7.2.65 M_GTIM - Acquire System Date/Time in Any Format

The M_GTIM service acquires the system date and time in any one of the three standard formats described in Appendix H. The user can also get the system date/time by using any of the three specific case macros. These macros generate the same SVC call but the function code is provided by the macro.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.GTIM.

Entry Conditions

Calling Sequence

M_GTIM [FUNCT=]*funct*, [TIMBUF=] *timbuf*

(or)

```
LA      R1,timbuf
ORMW    R1,funct
SVC     2,X'50'   (or)  M_CALL   H.REXS,74
```

funct is the address of a word containing the function code (see chart below) in byte 0 (most significant) and zeros in bytes 1, 2, and 3

timbuf is the address of a buffer where the service places the date and time in the format requested by the user. This buffer is 2 or 4 words in length depending on the format desired.

Function Code	Return Format	Buffer Length
1	Binary	2W
2	Byte binary	2W
3	Quad ASCII	4W

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 byte 0 contains the function code and bytes 1-3 contain the buffer address as used by the call. All others are returned intact.

Abort Cases

RX13 FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT

7.2.66 M_GTSAD - Get TSA Start Address

The M_GTSAD service returns the TSA starting address of the calling task.

This service is callable by the IPU.

The nonbase mode equivalent service is M.GTSAD.

Entry Conditions

Calling Sequence

M_GTSAD

(or)

SVC 2,X'7D' (or) M_CALL H.REXS,91

Exit Conditions

Return Sequence

M.IPURTN R1

Registers

R1 logical address of the TSA of the caller

7.2.67 M_HOLD - Program Hold Request

The M_HOLD service makes the specified task ineligible for CPU control by setting the hold bit in the CPU dispatch queue. The specified task remains in the hold state until the operator issues the OPCOM CONTINUE directive. If the specified task is not in the CPU dispatch queue, the request is ignored.

The nonbase mode equivalent service is M.HOLD.

Entry Conditions

Calling Sequence

M_HOLD [TASKID=]*task*

(or)

```
LI    R6,0          } (or) LD R6,taskname
LW    R7,taskno     }
SVC   1,X'58   (or)   M_CALL   H.REXS,25
```

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, contains the task number.

7.2.68 M_ID - Get Task Number

The M_ID service allows the user to pass the address of a parameter block containing any of the following: task number, task name, owner name, or task pseudonym. The service will provide the missing items if a matching entry is found. Initially, the caller passes zero as the index value following the parameter block address. If more than one task in the dispatch queue satisfies the given parameters, the service returns to the caller with an index value in R5 for retrieval of further entries. The caller is responsible for updating the index with the contents of R5 and reissuing M_ID until all tasks that meet specifications have been identified or R5 equals zero.

The nonbase mode equivalent service is M.ID.

Entry Conditions

Calling Sequence

M_ID [PBADDR=] *pbaddr*, [INDEX=] *index*

(or)

```
LW   R5,a variable equal to 0 or an index
LA   R7,pbaddr
SVC  1,X'64'  (or)  M_CALL   H.REXS,32
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

Word	Contents
0	task activation sequence number
1-2	task name
3-4	owner name
5-6	pseudonym

The user supplies those items that are known and zeros the other words.

index is a variable equal to zero for initial call, then previous DQE address for each subsequent call

Exit Conditions

Return Sequence

M.RTRN 5

Registers

Normal Return:

R5 bit 0 is set if more than one task satisfies the given parameters. Bits 1-31 contain the DQE address of the first matching task found. If no entry satisfies the given parameters, R5 equals zero. R5 may be used as input for subsequent calls.

Abnormal Return:

CC1 set parameter block address is invalid; R5 remains unchanged

7.2.69 M_INCLUDE - Include Shared Image

The M_INCLUDE service allows a base mode task to include a shared image or a memory partition into its address space. A static or dynamic partition can be included at the logical address specified at creation or at a logical address specified in the caller notification packet (CNP). The task is suspended until the inclusion is complete. If the resource was not included by another task, an allocated resource table (ART) and a shared memory table (SMT) entry are established for the resource. The resource is automatically allocated for explicit shared use. If inclusion is successful, the assign and user counts are incremented for the resource.

The shared image is identified by resource pathname or a resource identifier (RID) that was defined when the partition was created. A partition is identified by an 8-character partition name, or a resource identifier (RID) that was defined when the partition was created. If the partition is dynamic, it is identified by an 8-character owner name that is used to associate this copy of the partition with a particular set of users.

Prezeroing of partitions is not performed by this service. The resource is swappable with the task if the user count goes to zero, and remains allocated until the assign count is zero.

The option is provided to lock the resource for exclusive use. However, the resource remains locked until: the owner of the lock terminates, the Release Exclusive Lock (M_UNLOCK) service is explicitly called, or the resource is excluded by the task.

Once a shared image has been included by a task, subsequent includes by that task are ignored.

The equivalent nonbase mode service is M.INCLUDE.

Note: To ensure proper inclusion, the first eight characters of a shared image file name or partition name should be unique within the system.

Entry Conditions

Calling Sequence

M_INCLUDE [RESOURCE=] *addr* [,[CNPADDR=]*cnpaddr*]
(or)
LW R1, *addr*
LA R7, *cnpaddr* (or) ZR R7
SVC 2,X'40' (or) M_CALL H.REMM,12

addr contains a PN vector, an PNB vector, or an RID vector

cnpaddr is the address of a caller notification packet (CNP) if notification is desired or if a logical address for partition inclusion is specified

Applicable portions of the CNP for this function are time-out value, abnormal return address, option field, and status field.

The option field has the following significance:

Bit	Meaning if Set
0	read/write access
1	use written back data section
2	set exclusive resource lock
3	reserved for MPX-32
4	logical address of partition is supplied in word 4
5-15	reserved

Registers

R1	contains *addr*
R7	contains *cnpaddr*; otherwise zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R3, R5	M.RTRN R3,R5
(or)	(or)
M.RTRNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R3 contains the starting logical address of the shared image

R5 contains the allocation index which can be used to identify the shared image for the resource lock and image exclusion services. It contains the nonzero bias SMT index in the first byte and the address of the associated ART entry in the next three.

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set Posted in R7 or the status field of the CNP:

Value	Description
1	unable to locate shared image file
2	requested access mode not allowed
8	unrecoverable I/O error to volume
16	memory requirements conflict with task's address space
35	resource is not a shared image
36	requested physical memory already allocated
37	nonpresent physical memory requested
38	time out occurred waiting for shared memory to become available
40	invalid load module
41	invalid requested physical memory
44	write back requested and shared image has no write back section
45	loading error during inclusion of read only section of shared image
47	loading error during inclusion of read and write section of shared image
48	incompatible load addresses for shared image
49	excessive multicopied shared images with no read-only section (only three are allowed)
55	allocated resource table (ART) is full
58	shared memory table (SMT) space unavailable
62	resource specified is not a static or dynamic partition and the option field indicates a logical inclusion address is given
80	shared image version level is not compatible with executable image
98	requires more shadow memory than exists

7.2.70 M_INQUIRER - Resource Inquiry

The M_INQUIRER service obtains information specific to a resource allocated by a base mode task. The information is returned as pointers to the various data structures within the system that describe the resource. The resource must have been previously allocated or included for memory partitions by the caller. Resources are identified by a logical file code (LFC) obtained when the resource was allocated, a memory partition name defined when the partition was created, or an allocation index obtained when the resource was allocated or included. If not supplied as an argument, the caller is provided with the unique allocation index which can be used to set and release exclusive or synchronous locks on the resource while it remains allocated.

It is the caller's responsibility to interpret the information in the identified structures. This should be done by a user-supplied subroutine that acts as a common interface between application programs and this service. In this way, resource inquiries will be less sensitive to changes in system structures.

The nonbase mode equivalent service is M.INQUIRY.

Entry Conditions

Calling Sequence

M_INQUIRER [BUFFER=] *addr1*, [RESOURCE=] *addr2* [,[CNPADDR=] *cnpaddr*]

(or)

```
LA   R1, addr1
LD   R4, addr2
LA   R7, cnpaddr or ZR R7
SVC  2,X'48'   (or)   M_CALL   H.REMM,27
```

addr1 is the address of an 8-word parameter description area where the pointers to the appropriate system structure entries corresponding to this resource are to be returned

addr2 is the address of a doubleword containing zero in byte 0 and a 1- to 3-character, left-justified, blank-filled LFC in bytes 1, 2, and 3 of word 0 with zero in word 1

(or)

the address of a doubleword containing a 1- to 8-character, left-justified, blank-filled memory partition name in word 0 and the left-justified task number or address of a 1- to 8-character (left-justified, blank-filled) owner name in word 1

(or)

the address of a doubleword containing zero in word 0 and the allocation index obtained when the resource was assigned in word 1

cnpaddr is the address of a caller notification packet (CNP) if notification is desired. Applicable portions of the CNP for this function are abnormal return address and status field.

Registers

R1	contains *addr1*
R4	contains *addr2*
R7	contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R5	M.RTRN R5
(or)	(or)
M.RTNA R5 (CC1 set)	M.RTRN R5,R7 (CC1 set)

Registers

R5	contains the allocation index if not supplied as an argument, or zero if resource is undefined
R7	contains the return status if a CNP is not supplied; otherwise, unchanged

The interpretation of each word in the parameter description area and other pertinent information that can be extracted from each structure follows:

Word 0 - Allocated resource table (ART) address:

- number of tasks assigned to this resource
- number of tasks currently using resource
- exclusive lock owner (DQE index)
- synchronous lock owner (DQE index)
- current allocation usage mode
- current allocation access mode (implicit shared)
- shared relative EOF block number (implicit shared)
- shared relative EOM block number (implicit shared)

Word 1 - File assignment table (FAT) address:

- relative EOF block
- relative EOM block
- number of segments in file
- current segment number
- current access mode
- relative file block position
- volume number (unformatted media only)
- unformatted ID (unformatted media only)
- assigned access restrictions
- file attribute and status flags

Word 2 - Unit definition table (UDT) address:

- device type code
- logical channel number
- logical subchannel number
- physical channel number (if different from logical)
- physical subchannel number (if different from logical)
- sectors per block (disk/floppy)
- sectors per allocation unit (disk/floppy)
- sectors per track (disk/floppy)
- number of heads (disk/floppy)
- total number of allocation units (disk/floppy)
- sector size (disk/floppy)
- characters per line (TTY/terminal)
- lines per screen (TTY/terminal)
- tab size (TTY/terminal)
- tab settings (TTY/terminal)

Word 3 - Device type table (DTT) address:

- number of controller entries for device
- ASCII device mnemonic
- device type code

Word 4 - Controller definition table (CDT) address:

- controller I/O class
- number of devices on controller
- device type code
- interrupt priority level
- logical channel number
- logical subchannel number of first device
- address of interrupt handler
- interrupt vector location
- controller definition flags

Word 5 - Shared memory table (SMT) address (applies to memory partitions and shared images):

Extractable information:

- starting map register number
- memory type
- starting page number
- total number of pages
- number of map image descriptors
- address of map image descriptor list

Word 6 - File pointer table (FPT) address:

- logical file code associated with resource

Word 7 - Mounted volume table (MVT) address (applies only to volume resources):

- volume name
- current number of users of volume
- volume definition flags
- root directory resource ID
- number of descriptors available on volume
- number of allocation units available
- volume access restrictions

A value of zero returned in any word of the parameter description area implies the corresponding structure does not apply to the resource for which the inquiry was made. For example, only words 0 and 5 apply for memory partitions or shared images.

For volume resources, words 2 through 4 pertain to the device upon which the volume is mounted.

The MPX-32 Technical Manual Volume I contains a complete description of the various system structures.

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
5	shared memory table entry not found for partition
10	illegal address range
29	logical file code not assigned
30	invalid allocation index

7.2.71 M_INT - Activate Task Interrupt

The M_INT service allows the calling task to cause the previously declared break/task interrupt receiver routine of the specified task to be entered.

The nonbase mode equivalent service is M.INT.

Entry Conditions

Calling Sequence

M_INT [TASK=] *task*

(or)

```
ZR   R6                  }
LW   R7,taskno           }  (or)  LD R6,taskname
SVC  1,X'6F'   (or)  M_CALL  H.REXS,47
```

task the address of a doubleword containing the name of the task or 0 in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of 0 specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 unchanged if R7 is zero. If R7 is not zero, bit 0 of R6 is one if the specified task was not set up to receive a pseudointerrupt; otherwise bit 0 is zero. Bits 1-31 of R6 are zero in all cases.

R7 zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, contains the task number

7.2.72 M_IPUBS - Set IPU Bias

The M_IPUBS service allows the user to dynamically change the IPU bias state for the current task.

The nonbase mode equivalent service is M.IPUBS.

Entry Conditions

Calling Sequence

M_IPUBS [BIAS=] *bias*

(or)

```
LW      R7,bias
SVC     2,X'5B'    (or)    M_CALL  H.REXS,82
```

bias is the IPU bias state requested as follows:

Value	Description
0	nonbiased task; can be executed by either the CPU or IPU
1	CPU only; can be executed only by the CPU
2	IPU bias; can be executed by either the CPU or IPU but is given priority status by the IPU

Exit Conditions

Return Sequence

M.RTRN R6,R7

Registers

R6 contains execution status as follows:

Value	Description
0	normal return
1	IPU is not configured in the system
2	IPU is currently marked off-line

R7 contains the IPU bias state of the task before this service was issued as follows:

Value	Description
0	nonbiased task
1	CPU only
2	IPU bias

7.2.73 M_LIMITS - Get Base Mode Task Address Limits

The M_LIMITS service returns the current task limits of a specified base mode task into sequential word locations of a specified buffer until all the limits are provided or the buffer is full. The values returned are the TSA base address, the stack lower bound, the stack upper bound, the read-only section low address, and the read/write section low address.

Entry Conditions

Calling Sequence

M_LIMITS [BUFFER=] *addr*, [NUMBER=] *number*

(or)

```
LA   R1, addr
LI   R4, number
SVC  2,X'5D'   (or)   M_CALL   H.REXS,84
```

addr is the word bounded address of a buffer where the task address limits are to be stored

number is the number of bytes allocated for the buffer

Registers

R1 contains *addr*

R4 contains *number*

Exit Conditions

Return Sequence

Normal Return:

M.RTRN

Abnormal Return:

M.RTRN R7

Registers

CC1 error condition

R7 contains the error condition as a hexadecimal value as follows:

Value	Description
257	destination buffer is in the operating system
258	destination buffer is in the TSA
259	invalid destination buffer address
260	buffer length not a word multiple

7.2.74 M_LOCK - Set Exclusive Resource Lock

The M_LOCK service allows a task to obtain exclusive allocation of a resource, as though it were nonshareable, for as long as the lock is owned. The resource must have been previously allocated (included for memory partitions), and is identified by either a logical file code, LFC (defined when the resource was assigned) or an allocation index (obtained when the resource was assigned or by a resource inquiry). The task may request immediate denial if the lock is not available, or wait for an indefinite or specified period of time. An exclusive resource lock may be obtained for any allocated resource that is not being shared by multiple tasks at the time of the call to this service.

The nonbase mode equivalent service is M.LOCK.

Entry Conditions

Calling Sequence

M_LOCK [ARGA=] *arga* [,[CNP=]*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr  (or)  ZR   R7
SVC  2,X'44'  (or)  M_CALL   H.REMM,23
```

arga is an address containing the allocation index obtained when the resource was assigned

(or)

an address containing the address of a file control block (FCB) which contains a LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC is not assigned
30	invalid allocation index
38	timeout occurred while waiting to become lock owner
46	unable to obtain resource descriptor lock (multiprocessor only)
50	resource is locked by another task
51	resource is allocated to another task

Wait Conditions

The task is placed in a wait state, as appropriate, if specified with the CNP.

7.2.75 M_LOGR - Log Resource or Directory

The M_LOGR service provides the base mode task with a convenient interface to locate the directory entry and resource descriptor for a single resource or for all the resources defined in a specified directory.

The caller must make a resource specification in the resource logging block (RLB). The log service will evaluate the resource specification and determine whether to log a single resource or all the resources defined in a directory. Some resource specifications are ambiguous and require the caller to specify additional information so the type of log function requested can be determined.

To log all the resources defined in a specified directory, the M_LOGR service must be called repeatedly until the last resource in the directory is logged. The user must reset bit 0 to zero in RLB.TYPE to indicate the first call. The operating system automatically changes the contents of bit 0 to one to indicate recall. Once all resources on the directory are logged, the operating system automatically resets bit 0 back to zero to indicate all resources have been logged.

Note: The M_LOGR system service does not search the memory resource descriptor table (MDT) for resource descriptors.

7.2.75.1 Resource Specifications for Pathnames

The caller can specify any valid pathname that is recognized by the Volume Management Module. The log service recognizes all valid pathname variations. However, some pathnames are ambiguous within the context of this service and require special considerations for the service to function with the expected results.

Specifically, pathnames that end with a directory specification are interpreted to mean log the contents of the directory. Directories can be logged as resources in two ways. The first is to supply a pathname that specifies the directory as a resource. This specification is not ambiguous. The second way is to supply a pathname that ends with a directory specification. This type of pathname is ambiguous and requires special handling.

Examples

The following type of pathname always logs the directory entry and resource descriptor for the specified resource.

@*volume* (*directory*) *resource*

The following type of pathname usually specifies to log the contents of the specified directory. The meaning of this pathname can be changed by setting the log single flag (RLB.LS) bit in the RLB flag word (RLB.INT). When the RLB.LS flag is set, the directory entry and resource descriptor for the specified directory are returned.

@*volume* (*directory*)

The following type of pathname means log the specified directory. The directory entry and resource descriptor for the specified directory are returned.

@*volume*^*directory*

7.2.75.2 Resource Specifications for Pathname Blocks

Pathname blocks are processed in the same manner as pathnames.

7.2.75.3 Resource Specifications for a Resource Identifier

When a resource identifier (RID) is furnished, the log service assumes the indicated resource is a directory and attempts to log the indicated resource as a directory.

7.2.75.4 Resource Specifications for a Logical File Code (LFC), FCB Address, or Allocation Index

When this type of resource specification is provided the log service makes the following assumptions:

- The implied file control block (FCB) is assigned to a directory.
- The implied FCB is opened.
- The buffer address in the FCB is the buffer to be used by the log service for locating directory entries.
- The transfer quantity in the FCB is the maximum size of the directory entry buffer.
- The FCB must be an extended FCB and must be opened in random access mode.
- The buffer is empty on the initial call and positions to the beginning of the directory and primes the supplied buffer. The directory is not read again until it is exhausted.

The caller should assign the directory in read mode so the directory can be searched by other users as it is being logged.

The nonbase mode equivalent service is M.LOGR.

Entry Conditions

Calling Sequence

M_LOGR [RLBADDR=] *addr1* [, [CNPADDR=] *addr2*]

(or)

```
LA   R2,addr1
LA   R7,addr2  (or)  ZR  R7
SVC  2,X'29'  (or)  M_CALL H.VOMM,10
```

addr1 is the address of the resource logging block (RLB)

addr2 is a CNP address or zero if not supplied

RLB Structure on Initial Call

	0 7	8 15	16 23	24 31
Word 0	PN vector or RID vector or zero (RLB.TGT). See Note 1.			
1	192-word buffer address or zero (RLB.BUFA). See Note 2.			
2-3	Reserved for system use			
4	Type (RLB.TYPE) See Note 3.	Zero (RLB.BOFF). See Note 3.		
5	Length. See Note 4.	Directory return buffer address (RLB.DIRA). See Note 4.		
6	User FCB address or zero (RLB.FCB). See Note 5.			
7	Flags. See Note 6.	Reserved (RLB.INT).		

Notes:

1. If the PN vector length and address specify a resource, only one item is logged. If the specification does not end with a resource, but with a directory, the entire directory may be logged by repeated calls. A call by RID vector implies the RID is for a directory and all entries may be logged. A value of zero implies the entire contents of the current working directory.
2. This address must be doubleword bounded if this field is zero, the RD is not returned.
3. The type value should be zero if the call is by PN vector (length and address) or zero to indicate working directory. Type should be one to indicate a call by RID. If all resources in a directory are to be logged, bit 0 of RLB.TYPE must be zero to indicate the first call.
4. This word contains the address of a buffer and its length in words. The buffer may be up to 16 words long. The log service will place the first *n* words of the logged directory entry into this buffer. This provides the user access to the file name and other attributes that exist only in the directory entry.
5. This service uses the system FCB by default. Phasing problems may occur, as the directory to be logged must be deassigned between calls if multiple entries are desired. In many cases, the impact of having an entry deleted just after it has been logged, or having an entry appear after that portion in the directory has been scanned, will be small or nonexistent. In other cases, such as saving files in a directory, it may be major. To prevent these problems, the address of a FCB that will be used to hold the directory while logging occurs may be provided.

6. Bits in this word are assigned as follows:

Bit	Description
0-1	reserved
2	if set, directory entry and resource descriptor for specified directory are returned (RLB.LS)
3	root directory
4	used on return to indicate whether resource was located (see description of RLB Structure on Return under Exit Conditions)
5-7	reserved

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTNA (CC1 set)	M.RTRN R7

Registers

R7 return status if a CNP is not supplied; otherwise CNP address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

RLB Structure on Return

	0 7	8 15	16 23	24 31
Word 0	PN vector or RID vector or zero (RLB.TGT)			
1	192-word buffer address or zero (RLB.BUFA)			
2	MVTE address (RLB.MVTE). See Note 1.			
3	Disk address of directory RD (RLB.RDAD). See Note 1.			
4	Type (RLB.TYPE) See Note 2.	Byte offset of entry (RLB.BOFF)		
5	Length	Directory return buffer address (RLB.DIRA)		
6	User FCB address or zero (RLB.FCB)			
7	Flags. See Note 3.	Reserved (RLB.INT)		

Notes:

1. When all resources in a directory are to be logged, RLB.MVTE and RLB.RDAD are used by the operating system as input after the first call.

2. The operating system automatically changes the contents of bit 0 in RLB.TYPE as follows:

Value	Description
0	all resources in the directory have been logged; do not recall this service
1	recall this service and log the next resource in the directory

3. Bits in this word are assigned as follows:

Bit	Contents
0-1	reserved
2	directory entry and resource descriptor for specified directory are returned
3	root directory
4	zero if resource was not located
5-7	reserved

7.2.76 M_MEM - Create Memory Partition

The M_MEM service creates permanent memory partition definitions. Permanent memory partition definitions are given names in directories and remain known to the operating system until explicitly deleted.

MPX-32 uses memory partition definitions to establish the relationship of named globally accessible areas of memory to the tasks that require them.

This service first allocates a resource descriptor and defines the memory requirements for the partition. Next, the attributes of the partition are recorded in the resource descriptor. Finally, the name of the partition is established in the indicated directory.

When a directory entry is established, the directory entry is linked to the resource descriptor for the partition. This link relates the name of the partition to the other attributes of the partition. Typical partition attributes are:

- name
- resource identifier (RID)
- protection attributes
- management attributes
- memory requirements

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.MEM.

Entry Conditions

Calling Sequence

M_MEM [[PNADDR=] *pnaddr*] [, [RCB=] *rcbaddr*] [, [CNP=] *cnpaddr*]

(or)

```
LW   R1,pnaddr
LA   R2,rcbaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'22' (or) M_CALL  H.VOMM,3
```

pnaddr	contains a PN vector or PNB vector
rcbaddr	is the RCB address (required)
cnpaddr	is a CNP address or zero if CNP is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.77 M_MOD - Modify Descriptor

The M_MOD service allows the owner of a resource to change or alter the protection or other resource management attributes of a resource. The owner can restrict or allow attributes with this mechanism.

This service limits which information in a descriptor can be changed. For example, the volume space occupied by the resource cannot be changed because it would allow the caller to violate the integrity of the volume on which the resource resides.

The caller can modify:

- protection fields of the descriptor
- accounting fields of the descriptor
- extension attribute fields of the descriptor
- user data field of the descriptor (words 160 through 175)
- shared image field of the descriptor

This service is the first part of a two step operation. The caller must read the resource descriptor into memory in order to modify it. Once read into memory, the resource descriptor is locked (for example, protected from access) until the caller writes the modified descriptor back to the volume with the M_REWRIT service. The caller must issue the rewrite before modifying another descriptor.

Only the resource owner or the system administrator can modify a resource descriptor. The format of the descriptor and the type of data to be modified must be known by the modifier.

The nonbase mode equivalent service is M.MOD.

Entry Conditions

Calling Sequence

M_MOD [[PNADDR=] *pnaddr*] [, [RD=] *rdaddr*] [, [CNP=] *cnpaddr*]

(or)

```
LW   R1,pnaddr
LA   R6,rdaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'2A' (or) M_CALL  H.VOMM,11
```

pnaddr	contains a PN vector, PNB vector, or RID vector
rdaddr	is an RD buffer address (doubleword bounded, 192W length)
cnpaddr	is a CNP address or zero if CNP is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 Return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.78 M_MODU - Modify Descriptor User Area

The M_MODU service allows users with write, update, append, or modify access to a resource to change or alter the user area of the resource descriptor of that resource.

The user can change all fields in the user area of the resource descriptor, however, only words 160 through 175 should be modified. Words 176 through 190 are used by some utilities. Word 191 of the resource descriptor is a reserved location, and any changes to this word are ignored.

This service is the first part of a two step operation. The caller first reads the user area of the resource descriptor into memory to modify it. The resource descriptor is then locked (protected from access) until the caller writes the modified user area back to the volume with the Rewrite Descriptor User Area (M_REWRTU) service. The caller then must issue the rewrite before modifying another descriptor or descriptor user area.

The nonbase mode equivalent service is M.MODU.

Entry Conditions

Calling Sequence

M_MODU [PNADDR=] *pnaddr* [,[UAADDR=] *uaaddr*] [,[CNP=] *cnpaddr*]

(or)

```
LW   R1,pnaddr
LA   R6,uaaddr
LA   R7,cnpaddr
SVC  2,X'31'    (or)    M_CALL    H.VOMM,26
```

pnaddr	contains a PN vector, PNB vector, or RID vector
uaaddr	is a user area buffer address (doubleword bounded, 32W length)
cnpaddr	is a CNP address or zero if CNP is not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	Return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.79 M_MOUNT - Mount Volume

The M_MOUNT service establishes a task or TSM environment as a user of a volume. If the volume requested is not physically mounted, M_MOUNT notifies the operator to mount the volume, and creates a mounted volume table (MVT) entry for the volume. This entry remains in memory as long as there are established users of the volume.

If the volume requested is already physically mounted, M_MOUNT attempts a logical mount.

For nonpublic volumes, M_MOUNT allocates a volume assignment table (VAT) entry within the user's TSA, provided that the requested usage classification is compatible. A request to mount a public or nonpublic volume that is already physically and logically mounted is ignored.

The nonbase mode equivalent service is M.MOUNT.

Entry Conditions

Calling Sequence

M_MOUNT [RRSADDR=] *addr1* [,[CNPADDR=] *addr2*]

(or)

```
LA   R1, addr1
LA   R7, addr2  (or)  ZR    R7
SVC  2,X'49'    (or)  M_CALL  H.REMM,17
```

addr1 is the address of an RRS entry (type 9). See Chapter 5 in the MPX-32 Reference Manual Volume I for a description of RRS entries.

addr2 is the address of a caller notification packet (CNP) if notification is desired. Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Registers

R1 contains *addr1*

R7 contains *addr2*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
1	invalid volume name
2	volume use not allowed to this user
6	volume assignment table (VAT) space not available
8	unrecoverable I/O error to volume
11	invalid RRS entry
13	mount device not in system
18	invalid mount device specified
20	unable to initialize volume (volume unsafe)
21	J.MOUNT run request failed
34	system administrator attribute required to mount public volume
38	time out occurred while waiting for resource
42	user requested abort of mount process
43	user requested hold on mount process
53	mount device unavailable
55	allocated resource table (ART) space not available for mount device
57	unable to mount volume for requested usage
59	mounted volume table (MVT) space not available
73	file overlap occurred; check system console
91	unable to mount volume due to pending physical dismount
93	unable to perform physical mount due to system shutdown in progress
94	J.MOUNT attempted to mount an unformatted disk volume

Abort Cases

```
RM42     USER REQUESTED ABORT OF MOUNT PROCESS
```

Wait Conditions

When the volume is unavailable (status values 50-63), the task is placed in a wait state, as appropriate.

7.2.80 M_MOVE - Move Data to User Address

The M_MOVE service moves an arbitrary number of bytes from a location in the user's logical address space to a location in the user's read only memory section or the user's read/write memory section. This service can be used to set traps or modify data in the user's memory sections. This service cannot be used in shared read-only memory sections.

The nonbase mode equivalent service is M.MOVE.

Entry Conditions

Calling Sequence

M_MOVE [BUFFER=] *inbuffer*, [DESTADDR=] *outbuffer*, [NUMBER=] *number*

(or)

```
LA    R1, inbuffer
LA    R2, outbuffer
LI    R4, number
SVC   2,X'62' (or) M_CALL H.REXS,89
```

inbuffer	is the byte address of the buffer to be moved
outbuffer	is the destination byte address
number	is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Return Sequence

Normal Return:

M.RTRN

Abnormal Return:

M.RTRN R7

Registers

CC1 set error condition

R7 contains the error condition as a hexadecimal value as follows:

Value	Description
256	invalid source buffer address
257	destination buffer is in the operating system
258	destination buffer is in the TSA
259	invalid destination buffer address

7.2.81 M_MYID - Get Task Number

The M_MYID service allows the user to obtain status on the currently executing task.

The nonbase mode equivalent service is M.MYID.

Entry Conditions

Calling Sequence

M_MYID [PBADDR=] *pbaddr*

(or)

```
ZR   R5
SBR  R5,0
LA   R7,pbaddr
SVC  1,X'64' (or)  M_CALL   H.REXS,32
```

pbaddr is the logical word address of the first location of a parameter block formatted as follows:

Word	Contents
0	task activation sequence number
1-2	task load module name
3-4	owner name
5-6	pseudonym
7-8	current working directory, truncated to the first eight characters
9	reserved
10	scheduling flags (DQE.USHF)

Exit Conditions

Return Sequence

M.RTRN 5

Registers

CC1 set parameter block address

R5,R7 unchanged

Abort Cases

RX32 INVALID DQE ADDRESS

7.2.82 M_OPENR - Open Resource

The M_OPENR service prepares a resource for logical I/O and defines the intended access mode for subsequent operations on the resource. Protection is provided for both the requestor and the resource against indiscriminate access. If appropriate, additional FAT information is posted at this time. A blocking buffer may be allocated if not previously specified, explicitly or implicitly, during allocation of the resource. However, if a user-supplied buffer is specified in the FCB, that buffer will be used and any previously allocated blocking buffer will be released. A mount message is issued as a result of this function when the I/O is to be performed to a device associated with unformatted media, and the message is not inhibited by user request or previous open on the resource by another user. An open request to a resource that is already opened in the same access mode is ignored.

The nonbase mode equivalent service is M.OPENR.

Entry Conditions

Calling Sequence

M_OPENR [FCBADDR=]*fcbaddr* [,[CNPADDR=] *cnpaddr*]

(or)

```
LA   R1, fcbaddr
LA   R7, cnpaddr  (or)  ZR   R7
SVC  2,X'42'      (or)  M_CALL  H.REMM,21
```

fcbaddr is the address of a file control block (FCB)

cnpaddr is the address of a caller notification packet (CNP) if notification is desired. The applicable items of the CNP for this function are time-out value, abnormal return address, option field, and status field.

The option field describes the access and usage with the following interpretation:

byte 0 contains an integer value representing the specific access for which the resource is being opened. An error will be returned for an invalid integer. The following values are valid:

Value	Description
0	open for default access
1	open for read
2	open for write (resource redefined)
3	open for modify
4	open for update
5	open for append
6-255	reserved

byte 1 indicates the usage under which the resource is to be opened with the following bit significance when set. Only one of bits 0 and 1 in byte 1 may be set. If set, any usage specified at the time the resource was assigned is overridden. This may result in a denial condition if the usage specified at open differs from that specified at assignment.

Bit	Meaning if Set
0	open for explicit shared use
1	open for exclusive use
2	open in unblocked mode (overrides any specification made at resource assignment)
3	open in blocked mode (overrides any specification made at resource assignment)
4	resource data blocked. If the file is actually written to in any access mode (append, modify, update, write), the data will be recorded as blocked in the resource descriptor at the time the file is closed, regardless of whether or not the I/O was actually performed in blocked mode.
5-7	reserved

If a CNP is not supplied or the specification in the option field is zero:

1. The resource is opened for read access for a volume resource or update access for a device unless only a specific access mode was allowed at assignment. In that case the resource will be opened for that access.
2. The usage will be implicit shared or that specified at resource allocation, whichever is appropriate.

Registers

R1 contains *fcbaddr*

R7 contains *cnpaddr*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
2	specified access mode not allowed
4	blocking buffer not available
9	invalid usage specification
25	random access not allowed for this access mode
27	resource already opened in a different access mode
28	invalid access specification
29	no LFC that matches FCB file code
38	time out occurred waiting for resource
46	unable to obtain resource descriptor lock (multiprocessor only)
54	unable to allocate resource for specified usage

Wait Conditions

If an access or usage specification made at open changes the nature of the resource allocation, the task may be placed in a wait state, as appropriate, if specified in the CNP.

7.2.83 M_OPTIONDWORD - Task Option Doubleword Inquiry

The M_OPTIONDWORD service provides the caller access to both the first and second program option words. The first option word, which is the same word as supplied by the M_OPTIONWORD service, resides in R7. The second word resides in R6.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.PGOD.

Entry Conditions

Calling Sequence

M_OPTIONDWORD

(or)

SVC 2,X'C0' (or) M_CALL H.REXS,95

Exit Conditions

Return Sequence

M.IPURTN 6,7

Registers

R6	contains the 32-bit second option word
R7	contains the 32-bit first option word

7.2.84 M_OPTIONWORD - Task Option Word Inquiry

The M_OPTIONWORD service provides the caller with the 32-bit task option word. This word is also called the program option word.

This service may be executed by the IPU.

The nonbase mode equivalent service is M.PGOW.

Entry Conditions

Calling Sequence

M_OPTIONWORD

(or)

SVC 1,X'4C' (or) M_CALL H.REXS,24

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains the 32-bit task option word

7.2.85 M_OSREAD - Physical Memory Read

The M_OSREAD service moves an arbitrary number of bytes from a location in the physical space of MPX-32 to a location in the calling task's logical address space.

The physical space of MPX-32 includes:

- nonextended MPX-32 — location 0 through the end of memory pool
- extended MPX-32 — the beginning of extended MPX-32 through the end of the last map block of extended MPX-32

The nonbase mode equivalent service is M.OSREAD. This service is executable by the IPU.

Entry Conditions

Calling Sequence

M_OSREAD [[BUFFER=] *inbuffer*] [,[DESTADDR=] *outbuffer*]
[,[NUMBER=] *number*]

(or)

```
LA   R1,inbuffer
LA   R2,outbuffer
LI   R4,number
SVC  2,X'7E'     (or)  M_CALL  H.REXS,93
```

inbuffer is the 24-bit pure address of the byte bounded source buffer (in the physical space of MPX-32)

outbuffer is the 24-bit pure address of the byte bounded destination buffer (in the task's logical address space)

number is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Normal Return:

M.IPURTN

Abnormal Return:

M.IPURTN R7

Registers

CC1 set	error condition
R7	error condition as a decimal value:

Value	Description
256	invalid source buffer address
259	invalid destination buffer address
261	invalid number of bytes to be moved

7.2.86 M_OSWRIT - Physical Memory Write

The M_OSWRIT service moves an arbitrary number of bytes from a location in the calling task's logical address space to a location in the physical space of MPX-32.

The physical space of MPX-32 includes:

- nonextended MPX-32 — location 0 through the end of memory pool
- extended MPX-32 — the beginning of extended MPX-32 through the end of the last map block of extended MPX-32

This service is available only to privileged users and is executable by the IPU. The nonbase mode equivalent service is M.OSWRIT.

Entry Conditions

Calling Sequence

M_OSWRIT [[BUFFER=] *inbuffer*] [,[DESTADDR=] *outbuffer*]
[,[NUMBER=] *number*]

(or)

```
LA   R1,inbuffer
LA   R2,outbuffer
LI   R4,number
SVC  2,X'AF'   (or)   M_CALL   H.REXS,94
```

inbuffer is the 24-bit pure address of the byte bounded source buffer (in the task's logical address space)

outbuffer is the 24-bit pure address of the byte bounded destination buffer (in the physical space of MPX-32)

number is the number of bytes to be moved

Registers

R1	contains *inbuffer*
R2	contains *outbuffer*
R4	contains *number*

Exit Conditions

Normal Return:

M.IPURTN

Abnormal Return:

M.IPURTN R7

Registers

CC1 set — error condition

R7 — error condition as a decimal value:

Value	Description
256	invalid source buffer address
259	invalid destination buffer address
261	invalid number of bytes to be moved

7.2.87 M_PNAMB - Convert Pathname to Pathname Block

The M_PNAMB service converts a pathname to a form that can be analyzed by software. In most cases, utility programs use this to check the syntax of a pathname in a directive line. When called, this service parses the input pathname. If any errors are detected in the pathname syntax, this service terminates and updates R1 to indicate the point where the error was detected.

The nonbase mode equivalent service is M.PNAMB.

Entry Conditions

Calling Sequence

M_PNAMB [[PNADDR=] *pnaddr*] , [[PNB=] *pnbaddr*] [,[CNP=] *cnpaddr*]

(or)

```
LW   R1,pnaddr
LW   R4,pnbaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'2E' (or) M_CALL  H.VOMM,15
```

pnaddr contains a PN vector

pnbaddr contains a PNB vector

cnpaddr is a CNP address or zero if CNP is not supplied. Applicable portions of the CNP for this function are time-out value, abnormal return address, option field, and status field.

The option field of the CNP has the following interpretation:

Byte 0 is reserved

Byte 1 has the following significance when set:

Bit	Meaning if Set
0-6	reserved
7	parsing of the pathname includes only the volume and directory portions of the supplied pathname. This bit is usually set by J.TSM.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R1,R4	M.RTRN R1,R4
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R1	address of first PN character not processed and remaining length
R4	PNB address and actual PNB length
R7	return status if a CNP is not supplied; otherwise unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.88 M_PRIL - Change Priority Level

The M_PRIL service is provided to the privileged caller to dynamically alter the priority level of the specified task. Valid priority levels for real-time tasks are 1-54 inclusive. Valid priority levels for time distribution tasks are 55-64 inclusive. A real-time task cannot be changed to a time distribution priority level and a time distribution task cannot be changed to a real-time priority level. I/O continues to operate at base priority level of the cataloged task.

The nonbase mode equivalent service is M.PRIL.

Entry Conditions

Calling Sequence

M_PRIL [TASK=] *task* , [PRTY=] *priority*

(or)

```
LW   R5,priority
ZR   R6,=0          } (or) LD R6,taskname
LW   R7,taskno      }
SVC  1,X'4A'  (or)  M_CALL   H.REXS,9
```

task the address of a doubleword containing the name of the task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

priority is the priority level to be assigned to the task (1-54 for a real-time task; 55-64 for a time-distribution task)

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if the specified task was not found; otherwise, contains the task number

Abort Cases

RX06 UNPRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY LEVEL, OR A PRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY TO A LEVEL OUTSIDE THE RANGE OF 1 TO 64, INCLUSIVELY

7.2.89 M_PRIVMODE - Reinstate Privilege Mode to Privilege Task

The M_PRIVMODE service allows a task that was linked as privileged to return to a privileged status. See the M_UNPRIVMODE service.

The nonbase mode equivalent service is M.PRIV.

Entry Conditions

Calling Sequence

M_PRIVMODE

(or)

SVC 2,X'57' (or) M_CALL H.REXS,78

Exit Conditions

Return Sequence

M.RTRN

Status

CC1 set successful operation

7.2.90 M_PTSK - Parameter Task Activation

The M_PTSK service overrides specific task parameters in the load module or executable image preamble during activation. For unprivileged callers, some parameters are overridden by those of the calling task. The task name, optional resource requirements, and optional pseudonym are specified to the service call. When a task name is supplied in words 2 and 3 of the parameter task activation block (PTASK), the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied.

Options 1-32 reside in task option word 1. Options 33-64 reside in task option word 2. If using options 33-64, the expanded PTASK block format must be used and bit 4 of PTA.FLG2 must be set.

The nonbase mode equivalent service is M.PTSK.

Entry Conditions

Calling Sequence

M_PTSK [ACTADR=] *actaddr* [, [PSB=] *psbaddr*]]

(or)

```
LA   R1,actaddr
LA   R2,psbaddr (or) ZR R2
SVC  1,X'5F'  (or) M_CALL H.REXS,40
```

actaddr is the logical word address of the parameter task activation block. Words 0 to 12 of the block are required if options 1 through 32 are set; variable length RRS entries beginning in word 13 are optional. Words 0 to 19 are required if options 33 through 64 are set and bit 4 of PTA.FLG2 is set; variable length RRS entries beginning in word 20 are optional. The structure of the PTASK block follows.

psbaddr is the logical address of the parameter send block (PSB) or zero if no parameters are to be passed. If a load module name is supplied in the PSB, the load module must be in the system directory. A pathname vector or RID vector must be supplied if a load module is to be activated from a user directory.

The following is the structure of the expanded parameter task activation block:

Byte	Word	0 7	8 15	16 23	24 31
0	0	PTA.FLAG	PTA.NRRS	PTA.ALLO	PTA.MEMS
4	1	PTA.NBUF	PTA.NFIL	PTA.PRIO	PTA.SEGS
8	2-3	PTA.NAME			
10	4-5	PTA.PSN			
18	6-7	PTA.ON			
20	8-9	PTA.PROJ			
28	10	PTA.VAT	PTA.FLG2	PTA.EXTD	
2C	11	PTA.PGOW			
30	12	PTA.TSW			
34	13	PTA.RPTR			
38	14	PTA.PGO2			
3C	15	PTA.FSIZ		PTA.RSIZ	
40	16-19	Reserved (zero)			
50-*nn*	20-*nn*	RRS List			

Byte (Hex)	Symbol	Description
0	PTA.FLAG	contains the following:

Bit	Contents
0	reserved
1	job oriented (PTA.JOB)
2	terminal task (PTA.TERM)
3	batch task (PTA.BTCH)
4	debug overlay required (PTA.DOLY)
5	resident (PTA.RESD)
6	directive file active (PTA.DFIL)
7	SLO assigned to SYC (PTA.SLO)

For unprivileged callers, bits 0-3 are not applicable. These characteristics are inherited from the parent task.

1	PTA.NRRS	number of resource requirements or zero if same as summary entries in the load module or executable image preamble

Byte (Hex)	Symbol	Description
2	PTA.ALLO	memory requirement: number of 512-word pages exclusive of TSA, or zero if same as the preamble
3	PTA.MEMS	memory class (ASCII E, H or S) or zero if memory class is to be taken from the preamble. If the memory class is to be taken from the preamble, the caller has the option of specifying the task's logical address space in this field as follows: Bits — Contents 0-3 — hexadecimal value 0 through F representing the task's logical address space in megabytes where zero is 1MB and F is 16MB 4-7 — zero
4	PTA.NBUF	the number of blocking buffers required or zero if same as the preamble
5	PTA.NFIL	the number of FAT/FPT pairs to be reserved or zero if same as the preamble
6	PTA.PRIO	the priority level at which the task is to be activated or zero for the cataloged load module priority. See the Parameter Send Block section in Chapter 2 of this manual for more details.
7	PTA.SEGS	the segment definition count or reserved (zero)
8	PTA.NAME	contains the load module or executable image name, left justified and blank filled, or word 2 is zero and word 3 contains a pathname vector or RID vector
10	PTA.PSN	contains the 1- to 8-character ASCII pseudonym, left justified and blank filled, to be associated with the task or zero if no pseudonym is desired. For unprivileged callers, this attribute is inherited from the parent task if zero is supplied or the parent is in a terminal or batch job environment.
18	PTA.ON	contains the 1- to 8-character ASCII owner name, left-justified and blank-filled, to be associated with the task or zero if the task to default to the current owner name. Valid only when task has system administrator attribute.
20	PTA.PROJ	contains the 1- to 8-character ASCII project name, left-justified and blank-filled, to be associated with files referenced by this task, or zero if same as LMIT
28	PTA.VAT	the number of volume assignment table (VAT) entries to reserve for dynamic mount requests or zero if same as the preamble

Byte (Hex)	Symbol	Description
29	PTA.FLG2	contains the following flags: Bit — Meaning if Set 0 — debug activating task (PTA.DBUG) 1 — Command Line Recall and Edit is in effect for the task (PTA.CLRE) 2-3 — reserved 4 — expanded PTASK block flag (must be set to use options 33-64) (PTA.EBLK) 5 — reserved 6 — NOMAPOUT option (PTA.NMAP) 7 — MAPOUT option (PTA.MAP)
2A	PTA.EXTD	contains the following values: Bit — Meaning if Set -1 — maxaddr of extended MPX-32 -2 — minaddr of extended MPX-32 0 — extended MPX-32 is not specified *n* — a positive number representing a map block of MPX-32
2C	PTA.PGOW	contains the initial value of the task option word or zero
30	PTA.TSW	contains the initial value of the task status word or zero
34	PTA.RPTR	contains a pointer to the resource requirement summary list or, if an expanded PTASK block is not used, the RRS list begins here (see RRS list description - byte 50)
38	PTA.PGO2	contains the initial value of the second task option word
3C	PTA.FSIZ	contains the length of the fixed portion of the PTASK block in bytes
3E	PTA.RSIZ	contains the number of bytes of the resource requirement summary
40	Reserved	
50		resource requirement summary list. Each entry contains a variable length RRS. The RRS list has up to 384 words. Each entry must be doubleword bounded. Each entry is compared with the RRS entries in the LMIT. If the logical file code currently exists, the specified LFC assignment will override the cataloged assignment, otherwise the special assignment will be treated as an additional requirement and merged into the list. If MPX-32 Revision 1.*x* format of the RRS is specified, it is converted to the format

Byte (Hex)	Symbol	Description
		acceptable for assignment processing by the Resource Management Module (H.REMM). See MPX-32 Revision 1.*x* Technical Manual for format of the RRS.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 equals zero if the service was performed

R7 contains the task number of the task activated by this service

(or)

R6 equals one if invalid attempt to multicopy a unique task

R7 task number of existing task with same name

(or)

R0 destroyed

R6

Value	Description
2	file specified in words 2 and 3 of the PTASK block not in directory
3	unable to allocate file specified in words 2 and 3 of the PTASK block
4	file is not a valid load module or executable image
5	DQE is not available
6	read error on resource descriptor
7	read error on load module
8	insufficient logical/physical address space for task activation
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task receiver queue full
16	invalid PSB address
17	RRS list exceeds 384 words
18	invalid RRS entry in parameter block

R7 contains zero if task not found

7.2.91 M_PUTCTX - Put User Context

The M_PUTCTX service overwrites the most recent user context that was previously saved in the TSA stack area. If control is transferred to the user before the context is stored again, this will be the context used. The values are loaded as words from the specified address, up to the last even word allocated in the block or until all context is provided.

If an attempt is made to modify the most significant byte of program status doubleword (PSD) 1, only the condition code values are changed.

Entry Conditions

Calling Sequence

M_PUTCTX [BUFFER=]*addr* , [NUMBER=]*number*

(or)

```
LA   R1, addr
LI   R4, number
SVC  2,X'71'   (or)   M_CALL   H.EXEC,42
```

addr is a word bounded logical address in memory whose context is returned

number is the number of bytes allocated for the context block

Registers

R1 contains *addr*

R4 contains *number*

Exit Conditions

Return Sequence

Normal Return Sequence:

M.RTRN

Abnormal Return Sequence:

M.RTRN R7

Registers

CC1 set error condition

R7 contains an error condition as a hexadecimal value as follows:

Value	Description
256	invalid source buffer address
260	buffer length not a word multiple

7.2.92 M_QATIM - Acquire Current Date/Time in ASCII Format

The M_QATIM service acquires the system date and time in ASCII format. The date and time are returned in a 4-word buffer, the address of which is contained in the call. See Appendix H for date and time formats.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.QATIM.

Entry Conditions

Calling Sequence

M_QATIM [TIMBUF=]*addr*

(or)

```
LA      R1,addr
ORMW    R1,=X'03000000'
SVC     2,X'50'  (or)  M_CALL   H.REXS,74
```

addr is the address of a 4-word buffer to contain the date and time

Exit Conditions

Return Sequence

M.IPURTN

Registers

R1 used by call; all others returned intact

Abort Cases

```
RX13    FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS
        OUT OF RANGE
```

7.2.93 M_RADDR - Get Real Physical Address

The M_RADDR service allows unprivileged tasks to determine the physical memory address associated with a given logical address.

The nonbase mode equivalent service is M.RADDR.

Entry Conditions

Calling Sequence

M_RADDR [[LOGADDR=]*logicaladdr*]

(or)

```
LA   R1, logicaladdr
SVC  1,X'0E'    or   M_CALL   H.REXS,90
```

logicaladdr is the logical address to be translated

Exit Conditions

Registers

R7 contains the physical address

7.2.94 M_RCVR - Receive Message Link Address

The M_RCVR service allows the caller to establish the address of a routine to be entered for the purpose of receiving messages sent by other tasks.

The nonbase mode equivalent service is M.RCVR.

Entry Conditions

Calling Sequence

M_RCVR [RCVRADR=]*recvaddr*

(or)

```
LA   R7,recvaddr
SVC  1,X'6B'  (or)  M_CALL   H.REXS,43
```

recvaddr is the logical word address of the entry point of the receive message routine in the user's task

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 contains zero if the receiver address was invalid; otherwise contains the receiver address

7.2.95 M_READ - Read Record

The M_READ service performs the following functions:

- provides special random access handling for disk files
- unblocks system files and blocked files
- reads one record into the buffer indicated by the transfer control word (TCW) in the FCB

The nonbase mode equivalent service is M.READ.

Entry Conditions

Calling Sequence

M_READ [FCBADDR=]*addr*
(or)
LA R1, *addr*
SVC 1,X'31' (or) M_CALL H.IOCS,3

addr is the FCB address. Appropriate transfer control parameters are defined in the TCW. See Chapter 5 in the MPX-32 Reference Manual Volume I for further details concerning the FCB word.

Registers

R1 contains *addr*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03 AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY

IO06 INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.

IO30 ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE

IO32 CALLING TASK HAS ATTEMPTED TO PERFORM A SECOND READ ON A '$' STATEMENT THROUGH THE SYC FILE

IO33 READ WITH BYTE GRANULARITY REQUEST MADE WITH NEGATIVE BYTE OFFSET

IO34 READ WITH BYTE GRANULARITY REQUESTS MADE WITHOUT SETTING RANDOM ACCESS BIT IN FCB

```
IO35     READ WITH BYTE GRANULARITY REQUESTS ARE VALID FOR
         UNBLOCKED FILES ONLY
```

Output Messages

Dismount/mount messages if EOT and multivolume magnetic tape.

7.2.96 M_READD - Read Descriptor

The M_READD service reads a resource descriptor for a specified resource. This service can examine the attributes of any volume resource. It is the responsibility of the caller to be familiar with the fields of the resource descriptor in order to determine the recorded information. This service should be called by a user-supplied subroutine that acts as a common interface between application programs and this service. In this way, application programs are less sensitive to changes in organization and content of these data structures.

The nonbase mode equivalent service is M.LOC.

Entry Conditions

Calling Sequence

M_READD [RESOURCE=]*addr1* , [RDADDR=]*addr2* [, [CNPADDR=]*addr3*]

(or)

```
LW   R1, addr1
LA   R6, addr2
LA   R7, addr3  (or)  ZR     R7
SVC  2,X'2C'    (or)  M_CALL H.VOMM,13
```

addr1 contains a PN vector, a PNB vector, an RID vector, an LFC, or an FCB address

addr2 is the address of a resource descriptor buffer, doubleword bounded and 192W in length

addr3 is a CNP address or zero if CNP not supplied

Registers

R1 contains *addr1*

R6 contains *addr2*

R7 contains *addr3*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN R4	M.RTRN R4
(or)	(or)
M.RTNA R2 (CC1 set)	M.RTRN R7,R2 (CC1 set)

Registers

R2	contains the address of the last PN item processed in an abnormal return
R4	contains the MVTE address for the volume specified
R7	contains the return status if a CNP is not supplied. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.97 M_RELP - Release Dual-Ported Disk/Set Dual-Channel ACM Mode

The M_RELP service applies to dual-port extended I/O disks and allows the privileged user to release a device from its reserved state. When issued to an ACM that has been SYSGENed as full-duplex, this service can be used to set the ACM from single-channel to dual-channel mode (applies to ACMs using the H.F8XIO handler only).

The nonbase mode equivalent service is M.RELP.

Entry Conditions

Calling Sequence

M_RELP [FCB=]*fcb*

(or)

```
LA   R1,fcb
SVC  1,X'27'  (or)  M_CALL  H.IOCS,27
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

7.2.98 M_RENAME - Rename File

The M_RENAME service changes the name of an existing permanent file. This service can move a file from one directory to another directory on the same volume.

When called, this service creates the new name of the file in the specified directory and then deletes the old name of the file from the specified directory.

The nonbase mode equivalent service is M.RENAM.

Entry Conditions

Calling Sequence

M_RENAME [FILEADDR=]*addr1* , [PNADDR=]*addr2* [, [CNPADDR=]*addr3*]

(or)

```
LW   R1, addr1
LW   R2, addr2
LA   R7, addr3  (or)  ZR     R7
SVC  2,X'2D'    (or)  M_CALL  H.VOMM,14
```

addr1	contains the old PN or PNB vector
addr2	contains the new PN or PNB vector
addr3	is a CNP address or zero if CNP is not supplied

Registers

R1	contains *addr1*
R2	contains *addr2*
R7	contains *addr3*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA R7 (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7	contains the return status if a CNP is not supplied; otherwise, contains the denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.99 M_REPLACE - Replace Permanent File

The M_REPLACE service replaces the data contents of an existing permanent file with the data contents of an existing temporary file. The permanent file retains its original directory entry and resource descriptor.

This service allows utility programs to change the data contents of a file without changing any of the file's other attributes. This service maintains the integrity of a file's resource identifier.

This service can be used on any permanent file. When this service completes, the temporary file is deallocated and deleted. An error condition is returned if the permanent file is allocated to another at the time of the service call and bit 0 of the CNP option field is not set.

This service should only be used on files with the fast access attribute. For files that do not have this attribute, the same functionality can be accomplished by using the Delete Resource (M_DELETER) service followed by the Change Temporary File to Permanent File (M_TEMPFILETOPERM) service.

The nonbase mode equivalent service is M.REPLAC.

Entry Conditions

Calling Sequence

M_REPLACE [RESOURCE=]*addr1* , [PATH=]*vector* [, [CNPADDR=]*addr2*]

(or)

```
LA   R1, addr1
LW   R2, vector
LA   R7, addr2  (or)  ZR   R7
SVC  2,X'30'    (or)  M_CALL  H.VOMM,23
```

addr1	is the FCB or LFC address of the temporary file
vector	is the pathname vector of the permanent file
addr2	is a CNP address or zero if CNP is not supplied

Registers

R1	contains *addr1*
R2	contains *vector*
R7	contains *addr2*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

CC1 set	error condition
R7	contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.100 M_RESP - Reserve Dual-Ported Disk/Set Single-Channel ACM Mode

The M_RESP service applies to dual-port extended I/O disks and allows the privileged user to reserve a device to the requesting CPU until such time as a release (M_RELP) is issued. When issued to an ACM that has been SYSGENed as full-duplex, this service can reset the ACM from dual-channel to single-channel mode (applies to ACMs using the H.F8XIO handler only).

The nonbase mode equivalent service is M.RESP.

Entry Conditions

Calling Sequence

M_RESP [FCB=]*fcb*

(or)

```
LA   R1,fcb
SVC  1,X'26'  (or)  M_CALL  H.IOCS,24
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

7.2.101 M_REWIND - Rewind File

The M_REWIND service performs the following functions:

- issues an end-of-file and purge if the file is a system or blocked file which is output active
- for system and blocked files, initializes blocking buffer control cells for subsequent access
- rewinds the file or device

The nonbase mode equivalent service is M.RWND.

Entry Conditions

Calling Sequence

M_REWIND [FCBADDR=]*addr*

(or)

```
LA   R1, addr
SVC  1,X'37'   (or)   M_CALL  H.IOCS,2
```

addr is the FCB address

Registers

R1 contains *addr*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09    ILLEGAL OPERATION ON THE SYC FILE
```

7.2.102 M_REWRIT - Rewrite Descriptor

The M_REWRIT service writes a modified resource descriptor back to a volume and releases the modify lock on the descriptor. This is the last step of a two step operation (the first step is M_MOD).

When this service is invoked, the indicated resource descriptor is read into an internal buffer. The fields that are allowed to be modified are copied from the user supplied resource descriptor buffer to the appropriate areas of the internal buffer. Upon successful modification of the resource descriptor in the internal buffer, the resource descriptor is written to the correct location on the volume and the modify lock is released.

The nonbase mode equivalent service is M.REWRIT.

Entry Conditions

Calling Sequence

M_REWRIT [[RD=]*rdaddr*] [, [CNP=]*cnpaddr*]

(or)

```
LA   R6,rdaddr
LA   R7,cnpaddr (or) ZR  R7
SVC  2,X'2B'  (or)  M_CALL  H.VOMM,12
```

rdaddr is the RD buffer address. This must be the same address supplied by the caller for use with the associated Modify Descriptor (H.VOMM,11) call; doubleword bounded, 192W length

cnpaddr is a CNP address or zero if CNP not supplied

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.103 M_REWRTU - Rewrite Descriptor User Area

The M_REWRTU service writes a modified resource descriptor back to a volume and releases the modify lock on the descriptor. This is the last step of a two step operation; the first step is M_MODU.

When this service is invoked, the indicated resource descriptor is read into an internal buffer. The data from the buffer supplied by the user is then copied to the appropriate areas of the internal buffer. Upon successful modification of the resource descriptor in the internal buffer, the resource descriptor is written to the correct location on the volume and the modify lock is released.

The nonbase mode equivalent service is M.REWRTU.

Entry Conditions

Calling Sequence

M_REWRTU [[UA=]*uaaddr*] [, [CNP=]*cnpaddr*]

(or)

```
LA   R6,uaaddr
LA   R7,cnpaddr  (or)  ZR  R7
SVC  2,X'32'     (or)  M_CALL  H.VOMM,27
```

uaaddr is the user area buffer address. This must be the same address supplied by the caller for use with the associated Modify Descriptor User Area (H.VOMM,26) call; doubleword bounded and 32W length.

cnpaddr is a CNP address or zero if CNP is not supplied

Entry Conditions

Calling Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise denial address. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.104 M_ROPL - Reset Option Lower

The M_ROPL service allows the calling task to reset the option lower bit. Use the M_SOPL (Set Option Lower) service to set the option lower bit.

The nonbase mode equivalent service is M.ROPL.

Entry Conditions

Calling Sequence

M_ROPL

(or)

SVC 2,X'78' (or) M_CALL H.TSM,14

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN (CC1 set)

Status

CC1 set call caused the option lower bit to be reset

7.2.105 M_RRES - Release Channel Reservation

If the specified channel has not been reserved by this task, the M_RRES service ignores the request to release the channel and returns to the task. If the channel has been reserved by this task, the channel reserve indication is removed from the CDT entry.

After releasing the reserved channel, if any requests had been queued while the channel was reserved, IOCS resumes I/O to the associated device.

This service is not applicable for extended I/O channels.

The nonbase mode equivalent service is M.RRES.

Entry Conditions

Calling Sequence

M_RRES [CHAN=]*channel*

(or)

```
LW   R1,channel
SVC  1,X'3B'  (or)  M_CALL  H.IOCS,13
```

channel specifies the channel number (hexadecimal). If LW, load bits 24 to 31 of R1.

Exit Conditions

Return Sequence

M.RTRN

7.2.106 M_RSML - Resourcemark Lock

The M_RSML service is called to lock the specified resourcemark. It is used in conjunction with the Unlock Resourcemark service (M_RSMU) by tasks to synchronize access to a common resource. For further description, see Chapter 2.

The nonbase mode equivalent service is M.RSML.

Entry Conditions

Calling Sequence

M_RSML [LCKID=]*lockid*,[[TOPT=]*timev*] [,[POPT=]P]

(or)

```
LI    R4,timev
ZR    R5
[SBR  R5,0]
LI    R6,lockid
SVC   1,X'19'  (or)  M_CALL   H.REXS,62
```

lockid is a numeric resourcemark index value, 33 to 64 inclusive

timev is a numeric value which specifies action to be taken if the lock is already set and is owned by another task:

Value	Description
+1	immediate denial return
0	wait until this task is the lock owner (default)
-*n*	wait until this task is the lock owner, or until *n* timer units have expired, whichever occurs first

P indicates that while this task is waiting to become lock owner, the swapping mode is to be set to swap this task only if a higher priority task is requesting memory space. Otherwise, the task is a swap candidate if any task is requesting memory.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 zero if the request was accepted, otherwise contains a request denial code:

Value	Description
1	lock index exceeds maximum range
2	lock index is less than minimum range
3	lock is owned by another task (and *timev* = +1)
4	lock is owned by another task, *timev* = *-n* and *n* timer units have elapsed

7.2.107 M_RSMU - Resourcemark Unlock

The M_RSMU service unlocks a resourcemark which has previously been locked by a call to the Resourcemark Lock (M_RSML) service. If any other tasks are waiting to lock the specified resourcemark, the highest priority waiting task becomes the new lock owner.

The nonbase mode equivalent service is M.RSMU.

Entry Conditions

Calling Sequence

M_RSMU [LCKID=]*lockid*

(or)

```
LI    R6,lockid
SVC   1,X'1A'   (or)   M_CALL   H.REXS,63
```

lockid is a numeric resourcemark index value, 33 to 64 inclusive

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 zero if the request was accepted, otherwise contains a request denial code:

Value	Description
1	lock index exceeds maximum range
2	lock index is less than minimum range
3	lock is not owned by this task

7.2.108 M_RSRV - Reserve Channel

M_RSRV is a privileged service. If the task is unprivileged or the channel has been reserved previously by another task, this service makes a denial return. If the channel has not previously been reserved, the task number is stored in the CDT or UDT entry to mark the reservation. If any requests are currently queued for this channel, suspend is invoked until completion of any I/O currently in progress is complete. The standard handler is then disconnected from the service interrupt (SI) level. After reserving a channel, the task must connect its own handler to the SI dedicated location.

This service is not applicable for extended I/O channels.

The nonbase mode equivalent service is M.RSRV.

Entry Conditions

Calling Sequence

M_RSRV [CHAN=]*channel*, [DENADR=]*denial*

(or)

```
LW   R1,channel
LA   R7,denial
SVC  1,X'3A'  (or)  M_CALL  H.IOCS,12
```

channel specifies the channel number (hexadecimal) in bits 24 to 32. If using LW, load channel number in R1.

denial is the user's denial return address

Exit Conditions

Return Sequence

M.RTRN normal return

(or)

M.RTNA 7 denial return

Abort Cases

```
IO14    UNPRIVILEGED USER ATTEMPTING TO RESERVE CHANNEL
```

7.2.109 M_SETERA - Set Exception Return Address

The M_SETERA service changes the destination address upon exit from an established exception handler. This service can only be called from within an exception handler established by the M_SETEXA system service.

Entry Conditions

Calling Sequence

M_SETERA [TASK=]*addr*

(or)

```
LA   R7, addr
SVC  2,X'79'   (or)  M_CALL  H.REXS,81
```

addr is the address where execution continues upon exit from the handler

Registers

R7 contains *addr* or zero if the execution is to continue from point of trap

Exit Conditions

Return Sequence

M.RTRN R7

Registers

Normal Return:

R7 contains previous value

Abnormal Return:

CC1 set error condition

R7 contains error condition (hexadecimal equivalent) as follows:

Value	Description
259	invalid destination address

Abort Cases

```
RX15    ATTEMPT TO SET EXCEPTION RETURN ADDRESS WHEN
        ARITHMETIC EXCEPTION NOT IN PROGRESS
```

7.2.110 M_SETEXA - Set Exception Handler

The M_SETEXA service establishes a task exception handler, changes the location of the current task exception handler, or deletes the current task exception handler in base mode.

Entry Conditions

Calling Sequence

M_SETEXA [TASKID=]*addr*

(or)

```
LA   R7, addr
SVC  2,X'5C'   (or)   M_CALL   H.REXS,83
```

addr is the task address of the exception handler

Registers

R7 contains *addr*

Exit Conditions

Return Sequence

M.RTRN R7

Registers

Normal Return:

R7 contains previous value

Abnormal Return:

CC1 set error condition

R7 contains the error condition as hexadecimal value as follows:

Value	Description
259	invalid destination buffer address

7.2.111 M_SETS - Set User Status Word

The M_SETS service allows the calling task to modify any task's user status word. Along with the Test User Status Word (M_TSTS) service, this is one of the means provided by MPX-32 for task-to-task communication. The user status word resides in the CPU dispatch queue (DQE.USW) and has a value of zero until modified by this service. The user status word is removed from the queue, modified as specified, and replaced in the queue.

The nonbase mode equivalent service is M.SETS.

Entry Conditions

Calling Sequence

M_SETS [FUNCT=]*function* , [STATW=]*statusw* [, [TASK=]*task*]

(or)

```
LD    R4,task
LI    R6,function
LW    R7,statusw
SVC   1,X'48'  (or)  M_CALL   H.REXS,7
```

function specifies the type of modification to be performed:

STF (1)	set flag
RSF (2)	reset flag
STC (3)	set counter
INC (4)	increment counter

If using the macro call, specify the alphabetic code. If loading registers, specify the corresponding numeric.

statusw contains a function parameter specific to function codes as follows:

Value	Description
1	bit position in the status word to be set (1 to 31)
2	bit position in the status word to be reset (1 to 31)
3	value to which the status word is to be set
4	value by which the status word is to be incremented

task is the address of a doubleword containing the name of the task, or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero or omission of the argument specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 5

Registers

R5 bit 0 set if the specified task was not found in the dispatch queue or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, zero

Abort Cases

RX05 INVALID FUNCTION CODE HAS BEEN SPECIFIED FOR REQUEST TO SET USER STATUS WORD

7.2.112 M_SETSYNC - Set Synchronous Resource Lock

The M_SETSYNC service is used in conjunction with the Release Synchronous Lock (M_UNSYNC) service for resource gating of explicitly shared resources where there is no automatic synchronization performed by the system. The mechanism allows a task to obtain synchronized access to a resource that has been concurrently allocated to multiple tasks. A synchronization lock can be obtained for any resource, provided it has been previously allocated (included for memory partitions) by the calling task. Unlike an exclusive lock, the synchronous lock does not prevent other tasks from allocating the resource in explicit shared mode. It is the sharing tasks' responsibility to synchronize access by cooperative use of the synchronous lock services. The resource is identified by either a logical file code (LFC), defined when the resource was assigned, or an allocation index, obtained when the resource was assigned or by a resource inquiry. If the synchronization lock is not available, the calling task can obtain an immediate denial return, or wait for an indefinite or specified period of time.

The nonbase mode equivalent service is M.SETSYNC.

Entry Conditions

Calling Sequence

M_SETSYNC [ARGA=]*arga* [, [CNP=]*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr  (or)  ZR   R7
SVC  2,X'46'  (or)  M_CALL   H.REMM,25
```

arga is an address containing the allocation index obtained when the resource was assigned

(or)

an address containing the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired

Applicable portions of the CNP for this function are time-out value, abnormal return address, and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC was not assigned by this task
30	invalid allocation index
38	timeout occurred while waiting to become lock owner
46	unable to obtain resource descriptor lock (multiprocessor only)
50	resource is locked by another task

Wait Conditions

The task is placed in a wait state, as appropriate, if specified by the CNP.

7.2.113 M_SETT - Create Timer Entry

The M_SETT service builds an entry in the timer table so that the requested function is performed upon time-out. Timer entries can be created to activate a program, resume a program, set a bit in memory, reset a bit in memory, or request an interrupt. Any task may create a timer to activate or resume a program. Timer entries to set or reset bits can be created by any task, provided the bit is within a static memory partition. Only privileged tasks can set bits in the operating system and request an interrupt.

The nonbase mode equivalent service is M.SETT.

Entry Conditions

Calling Sequence

M_SETT [TIMER=]*timer* , [SETVAL=]*t1* , [RSTVAL=]*t2* , [FUNCT=]*function*,
[ARG4=]*arg4* , [ARG5=] *arg5*

(or)

```
LB          R3,function
SLL         R3,24
ORMW        R3,timer
LW          R4,t1
LW          R5,t2
LW (or LD)  R6,arg4
(LW         R7,arg5)
SVC         1,X'45'  (or)  M_CALL H.REXS,4
```

timer is a word containing zeros in bytes 0 and 1, and a 2-character timer identification in bytes 2 and 3

t1 contains the current value to which the timer will be set in negative time units

t2 contains the value to which the timer will be reset upon each time-out in negative time units. If the reset value is zero, the function is performed upon time-out and the timer entry is deleted. This case is called a "one-shot" timer entry.

function specifies the function to be timed:

ACP (1)	activate program
RSP or RST (2)	resume program
STB (3)	set bit
RSB (4)	reset bit
RQI (5)	request interrupt

If using the macro call, specify the alphabetic code. If loading registers, specify the corresponding numeric.

The function and *arg4* and *arg5* contain values specific to the function being timed as follows:

Function Code		
Alphabetic	Numeric	*arg4* and *arg5*
ACP	1	*arg4* is a doubleword containing the 1- to 8-character name of the program to be activated (system file), or pathname vector or RID vector in the first half of the doubleword and zero in the second half of the doubleword. If the task named is not currently in execution, it is preactivated to connect the interrupt to the task. This connection remains in effect until the task aborts or the timer is deleted. On normal exit, the timer table is updated to point to the next generation. *arg5* is null.
RSP	2	*arg4* is a doubleword containing the 1- to 8-character name of the task to be resumed or zero in R6 and the task number in R7. *arg5* is null.
(or)		
RST	2	*arg4* is the task number entered into R7 and R6 is zeroed. *arg5* is null.
STB	3	*arg4* contains the address of the word in which the bits are to be set. The address must be within a static memory partition or the operating system. *arg5* contains the bit configuration of the mask word to be ORed.
RSB	4	*arg4* contains the address of the word in which the bit is to be reset. The address must be within a static memory partition or the operating system. *arg5* contains the bit configuration of the mask word to be ANDed.
RQI	5	*arg4* contains the priority level of the interrupt to be requested. *arg5* is null.

Normal Return Sequence:

M.RTRN R3

R3 is nonzero and condition codes are not set

Error Condition

M.RTRN R3

If there are no timer entries available, R3 is zero and condition codes are not set.

If there are timer entries available, R3 is zero and one of the following condition codes are set:

- CC1 set if requested load module does not exist or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file
- CC2 set if requested task is not active
- CC3 set if attempting to create a duplicate timer ID

Abort Cases

RX02 INVALID FUNCTION CODE SPECIFIED FOR REQUEST TO CREATE A TIMER ENTRY. VALID CODES ARE ACP(1), RSP OR RST(2), STB(3), RSB(4) AND RQI(5).

RX03 TASK ATTEMPTED TO SET/RESET A BIT OUTSIDE OF A STATIC PARTITION OR THE OPERATING SYSTEM

RX04 THE REQUESTING TASK IS UNPRIVILEGED OR HAS ATTEMPTED TO CREATE A TIMER ENTRY TO REQUEST AN INTERRRUPT WITH A PRIORITY LEVEL OUTSIDE THE RANGE OF X'12' TO X'7F', INCLUSIVE

7.2.114 M_SMSGR - Send Message to Specified Task

The M_SMSGR service allows a task to send up to 768 bytes to the specified destination task. Up to 768 bytes can be accepted as return parameters. For further description, see Chapter 2.

The nonbase mode equivalent service is M.SMSGR.

Entry Conditions

Calling Sequence

M_SMSGR [PSB=]*psbaddr*

(or)

```
LA   R2,psbaddr
SVC  1,X'6C'  (or)  M_CALL  H.REXS,44
```

psbaddr is the logical address of the parameter send block (PSB). See Chapter 2 for PSB description.

Exit Conditions

Return Sequence

M.RTRN 6

Registers

R6 contains the processing start (initial) error status if any:

Value	Description
0	normal initial status
1	task not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file
2-9	reserved
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task queue depth exceeded
16	invalid PSB address

7.2.115 M_SOPL - Set Option Lower

The M_SOPL service allows the calling task to set the option lower bit. Use the M_ROPL (Reset Option Lower) service to reset the option lower bit.

The nonbase mode equivalent service is M.SOPL.

Entry Conditions

Calling Sequence

M_SOPL

(or)

SVC 2,X'77' (or) M_CALL H.TSM,13

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN (CC1 set)

Registers

CC1 set call caused the option lower bit to be set

7.2.116 M_SRUNR - Send Run Request to Specified Task

The M_SRUNR service allows a task to activate or re-execute the specified destination task with a parameter pass of up to 768 bytes. Up to 768 bytes can be accepted as return parameters. For further description, see Chapter 2.

If a task activated with the TSM ACTIVATE directive is sent a run request, the queued run request is ignored. However, if a task is activated with a run request and a second run request is sent, the queued run request is then executed.

When a task name is supplied in words 0 and 1 of the parameter send block (PSB), the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied.

The nonbase mode equivalent service is M.SRUNR.

Entry Conditions

Calling Sequence

M_SRUNR [PSB=]*psbaddr*

(or)

```
LA   R2,psbaddr
SVC  1,X'6D'  (or)  M_CALL  H.REXS,45
```

psbaddr is the logical address of the parameter send block (PSB). See Chapter 2.

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6 contains the processing start (initial) error status if any:

Value	Description
0	normal initial status
1	reserved
2	file specified in the PSB not found in directory
3	reserved
4	file specified in the PSB is not a load module or executable image
5	dispatch queue entry (DQE) unavailable
6	I/O error on directory read
7	I/O error on load module read
8	memory unavailable

Value	Description
9	invalid task number for run request to multicopied load module in RUNW state
10	invalid priority
11	invalid send buffer address or size
12	invalid return buffer address or size
13	invalid no-wait mode end-action routine address
14	memory pool unavailable
15	destination task queue depth exceeded
16	invalid PSB address
17	reserved

R7 contains the task number of the destination task, or zero if the request was not processed

7.2.117 M_SUAR - Set User Abort Receiver Address

The M_SUAR service sets up an address to return control to if an abort condition occurs during task execution.

All files remain open prior to transferring to the user specified address. See Task Termination Sequencing in Chapter 2.

The nonbase mode equivalent service is M.SUAR.

Entry Conditions

Calling Sequence

M_SUAR [RCVADR=]*address*

(or)

```
LA   R7,address
SVC  1,X'60'  (or)  M_CALL  H.REXS,26
```

address is the logical address to which control will be transferred on task termination

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 bit 0 is zero if the request is honored, or one if the request is denied because the specified address is outside the user's allocated area; bits 1-31 are unchanged

Abort Cases

RX89 AN UNPRIVILEGED TASK HAS ATTEMPTED TO REESTABLISH AN ABORT RECEIVER (OTHER THAN M.IOEX)

7.2.118 M_SUME - Resume Task Execution

The M_SUME service resumes a task that has been suspended. A request to resume a task which is not suspended is ignored.

The nonbase mode equivalent service is M.SUME.

Entry Conditions

Calling Sequence

M_SUME [TASK=]*task*

(or)

```
ZR   R6          }
LW   R7,taskno   } (or) LD R6,taskname
SVC  1,X'53'  (or)  M_CALL  H.REXS,16
```

task the address of a doubleword containing the name of a task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared.

Exit Conditions

Return Sequence

M.RTRN R7

Registers

R7 zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, unchanged

7.2.119 M_SURE - Suspend/Resume

The M_SURE service suspends the calling task and resumes the target task. The suspend and resume functions are combined into one module providing faster context switching. It does not replace M_SUSP and M_SUME.

M_SURE applies to base mode realtime and time distribution tasks with the target task priority higher than or equal to that of the calling task. Context switch time can be further improved by turning real-time accounting off. This service is not recommended for two IPU biased tasks.

The accounting option is turned off using the OFRA option to the MODE directive in SYSGEN and OPCOM and the Catalog ENVIRONMENT directive.

The nonbase mode equivalent service is M.SURE.

Entry Conditions

Calling Sequence

M_SURE *taskno*

(or)

```
LW   R7,taskno
SVC  5,X'00'
```

taskno is the task number of the target task

Exit Conditions

Return Sequence

No return. All registers are destroyed. When the service completes normally, CC1 is reset. The next instruction is in the target task.

Abnormal Return:

CC1 set

R7 contains a code describing the reason for the error:

Value	Description
1	task not found
2	task not in suspend state
3	owner/access violation

Return is done via LPSD to the calling task.

7.2.120 M_SUSP - Suspend Task Execution

The M_SUSP service results in the suspension of the caller or any other specified task for the specified number of time units or for an indefinite time period, as requested. A task suspended for a time interval results in a one-shot timer entry to resume the task upon time-out of the specified interval. A task suspended for an indefinite time interval must be resumed through the M_SUME system service. Suspension of a task can also be ended upon receipt of a message interrupt. A message sent to a task that is synchronized (M_SYNCH) and suspended is not received, but the task is resumed.

The nonbase mode equivalent service is M.SUSP.

Entry Conditions

Calling Sequence

M_SUSP [TASK=]*task*, [TIM=]*time1*

(or)

```
LW   R5,time1
LI   R6,0          } (or)  LD R6,taskname
LW   R7,taskno     }
SVC  1,X'54'  (or)  M_CALL  H.REXS,17
```

task the address of a doubleword containing the name of a task or zero in word 0 and the task number in word 1. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

time1 contains zero, if suspension for an indefinite time interval is requested, else the negative number of time units to elapse before the caller is resumed.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 zero if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, contains the task number

(or)

zero and CC1 is set if the specified task name is multicopied

7.2.121 M_SYNCH - Set Synchronous Task Interrupt

The M_SYNCH service causes message and task interrupts to be deferred until the user makes a call to M_ANYWAIT, M_AWAITACTION, M_WAIT, or M_ASYNCH. When this service is used, message interrupts are not interrupted by end-action interrupts. All task interrupt levels cannot be interrupted, except by break, until they voluntarily relinquish control.

If a synchronized task is suspended then a message is sent to the task, the message receiver is not entered and the task resumes.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.SYNCH.

Entry Conditions

Calling Sequence

M_SYNCH

(or)

SVC 1,X'1B' (or) M_CALL H.REXS,67

Exit Conditions

Return Sequence

M.IPURTN

Registers

CC1 set synchronous task interrupt was already set

7.2.122 M_TBRKON - Trap Online User's Task

The M_TBRKON service processes a pause or break from the terminal or calling task. The service is also the default receiver for any online task and is called as a result of a hardware or software break. If a transfer control word (TCW) is specified, a user message is printed with the break message. Refer to the description of FCB Word 1 in Chapter 5 for more information.

The nonbase mode equivalent service is M.TBRKON.

Entry Conditions

Calling Sequence

M_TBRKON [TCW=]*tcw*

(or)

```
LW   R2,tcw   (or)   ZR R2
SVC  1, X'5C' (or)   M_CALL H.TSM,6
```

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
TS01     USER REQUESTED REMOVAL FROM A BREAK REQUEST
RX34     TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO
         BREAK IS ACTIVE
```

7.2.123 M_TDAY - Time-of-Day Inquiry

The M_TDAY service obtains the time-of-day as computed from the real-time clock interrupt counter. The counter is initialized with a SYSGEN parameter.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.TDAY.

Entry Conditions

Calling Sequence

M_TDAY

(or)

SVC 1,X'4E' (or) M_CALL H.REXS,11

Exit Conditions

Return Sequence

M.IPURTN 7

Registers

R7 Byte	Contents
0	hours (0 to 23)
1	minutes (0 to 59)
2	seconds (0 to 59)
3	interrupts (less than one second)

7.2.124 M_TEMPFILETOPERM - Change Temporary File to Permanent File

The M_TEMPFILETOPERM service makes a temporary file permanent. The temporary file is given a name in the specified directory and the file's resource type is changed from temporary to permanent. The file is made permanent with the attributes that were defined when it was created and with any new attributes that were acquired while the file's data was being established, such as additional extensions, end-of-file position, or explicit resource descriptor modifications incurred prior to invocation of this service. The temporary file can be made permanent only on the volume where the temporary file resides; for example, cross volume definitions are not allowed.

This service ensures exclusive use of a file while the initial file data is being established. The integrity of the file can be guaranteed before the file is defined in a directory where others can gain access to it.

When the directory entry is established, it is linked to the resource descriptor of the file. This link relates the name of the file to the other attributes of the file. These attributes are the same as the attributes for a permanent file.

The nonbase mode equivalent service is M.TEMPER.

Entry Conditions

Calling Sequence

M_TEMPFILETOPERM [RESOURCE=]*addr1* , [PNADDR=]*addr2*
[, [CNPADDR=]*addr3*]

(or)

```
LW   R1, addr1
LW   R2, addr2
LA   R7, addr3  (or)   ZR     R7
SVC  2,X'28'    (or)   M_CALL  H.VOMM,9
```

addr1 is an LFC or FCB address

addr2 contains a PN vector or PNB vector

addr3 is a CNP address or zero if CNP is not supplied

Registers

R1 contains *addr1*

R2 contains *addr2*

R7 contains *addr3*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.125 M_TRUNCATE - Truncate File

The M_TRUNCATE service allows the caller to truncate the unused space of a file. This service is the complement of the extend service (M_EXTENDFILE). Only files manually extended may need to be truncated.

This service truncates only temporary or permanent files. Directories and memory partitions cannot be truncated. The caller must have write, update or append access to truncate the file. A file cannot be truncated to less than the minimum space requirement of the file as defined when the file was created.

A file can be truncated regardless of whether it is currently allocated. Additionally, any allowable resource specification can be supplied, such as pathname (PN), pathname block (PNB), resource ID (RID), logical file code (LFC), or address of a file control block (FCB).

Asynchronous abort and delete are inhibited during execution of this service.

The nonbase mode equivalent service is M.TRNC.

Entry Conditions

Calling Sequence

M_TRUNCATE [RESOURCE=]*addr1* [, [CNPADDR=]*addr2*]
(or)
LW R1, *addr1*
LA R7, *addr2* (or) ZR R7
SVC 2,X'26' (or) M_CALL H.VOMM,7

addr1 contains a PN vector, PNB vector, RID vector, LFC, or FCB

addr2 is a CNP address or zero if CNP is not supplied

Registers

R1 contains *addr1*

R7 contains *addr2*; otherwise, zero

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 contains the return status if a CNP is not supplied; otherwise, unchanged. For return status codes, refer to the H.VOMM status codes in the Resource Assignment/Allocation and I/O chapter of Volume I.

7.2.126 M_TSCAN - Scan Terminal Input Buffer

The M_TSCAN service parses the line buffer pointed to by T.LINBUF. The service is used by tasks to scan a line of terminal input. The parameters (fields) to be scanned are in the user's line buffer. Each call to M_TSCAN returns one parameter from the line buffer and updates the current scan position. When a call returns a zero in R5 and a carriage return in R4, end of line (EOL) has been reached. Each read from the terminal reinitializes the line buffer and the pointer.

The nonbase mode equivalent service is M.TSCAN.

Entry Conditions

Calling Sequence

M_TSCAN

(or)

SVC 1,X'5B' (or) M_CALL H.TSM,2

Exit Conditions

Return Sequence

M.RTRN 4,5,6,7

(or)

M.RTRN CC1 set if a line buffer is not found.

Registers

R4	contains the delimiting character; carriage return if CC1 set
R5	number of significant characters before delimiter; zero if CC1 set
R6,7	first eight characters of the character string, left-justified. The entire character string is in words 0 through 3 of the terminal line buffer. Base mode tasks must locally define the value of T.LINBUF in order to access that field of the TSA.

Note: M_TSCAN ignores all blanks encountered before the first parameter or delimiter. If M_TSCAN encounters a delimiter before the first parameter, it continues to ignore all blanks until encountering the first parameter.

The M_RWND service resets the cursor at the first parameter of the current input line. M_TSCAN scans the line without any additional IOCS calls.

7.2.127 M_TSMPC - TSM Procedure Call

The M_TSMPC service receives TSM procedure call directive strings and returns the results of the directive or an error message to the user-supplied buffer. The service supports the following procedure call directives: $BATCH, $DIRECTORY, $ERR, $GETPARM, $LINESIZE, $PAGESIZE, $PROJECT, $RRS, $SET, $SETI, $TABS and $VOLUME.

The maximum size input string is 72 characters. The size of the output string depends on the input directive as follows:

Directive	Maximum Output
$BATCH	no output for normal processing
$DIRECTORY	16-character directory name from the M.KEY file
$ERR	212 characters and two carriage control characters (CR/LF) per line for an abort code definition. The ASCII control characters for LF and CR delimit the lines returned from $ERR.
$GETPARM	72 characters for a parameter value. If the specified parameter exists but has not received a value, the parameter name is returned. If the parameter does not exist, an error message is returned.
$LINESIZE	no output for normal processing
$PAGESIZE	no output for normal processing
$PROJECT	8-character project name from the M.KEY file
$RRS	variable length RRS entry for user-supplied LFC assignment
$SET/$SETI	no output for normal processing
$TABS	8 tab settings from the M.KEY file
$VOLUME	16-character volume name from the M.KEY file

Error messages are a maximum of 80 characters and two ASCII control characters (CR/LF) as EOL delimiters.

R7 must be zero on entry to this service.

Refer to the Notes section below for information on the syntax of the directives.

The nonbase mode equivalent service is M.TSMPC.

Entry Conditions

Calling Sequence

M_TSMPC *pcb*

(or)

```
LA   R1,pcb
ZR   R7
SVC  2,X'AE'   (or)    M.CALL H.TSM,17
```

pcb is the address of a 4-word procedure call block

Procedure Call Block (PCB)

The PCB contains the information necessary for the service to complete a procedure call. The format of the PCB is as follows:

	0 7	8 15	16 23	24 31
Word 0		Send buffer address (PCB.SBA)		
1	Send quantity (PCB.SQUA)			
2		Return buffer address (PCB.RBA)		
3	Actual return length (PCB.ACRP)		Return buffer length (PCB.RPBL)	

Send buffer address	is the address of a character string that represents a valid TSM procedure call directive
Send quantity	contains the length in bytes of the TSM procedure call directive
Return buffer address	is the address of a buffer to contain either valid return information or an error message if CC1 is set and R7 contains a value of 1
Actual return length	is the number of bytes returned from the procedure call
Return buffer length	is the size of the supplied return buffer

Exit Conditions

Return Sequence

M.RTRN

(or)

M.RTRN R7 (CC1 set)

Registers

R7 return status if error; otherwise, zeroed

Status

CC1 set

Posted in R7:

Value	Description
1	return buffer contains error message
2	invalid send buffer address
3	send buffer size is zero
4	send buffer too long
5	invalid return buffer address
6	return buffer size is zero
7	return data has been truncated
8	invalid PCB address
9	invalid SVC from a non-TSM task

Note: For the syntax of the $BATCH, $ERR, $LINESIZE, $PAGESIZE, $SET, and $SETI directives, refer to the MPX-32 Reference Manual Volume II, Chapter 1. The $RRS directive is similar to the $ASSIGN directive. Refer to the $ASSIGN directive syntax in the MPX-32 Reference Manual Volume II, Chapter 1 and specify $RRS rather than $ASSIGN. The syntax for $DIRECTORY, $PROJECT, $TABS, and $VOLUME is the directive name or its 4-character abbreviation plus the $ if used. These directives display information only. The syntax of the $GETPARM directive is as follows:

$GETPARM *parm*

parm is the name of a parameter defined in the directive file associated with the task. No percent sign (%) precedes the parameter name.

7.2.128 M_TSTE - Arithmetic Exception Inquiry

The M_TSTE service resets the arithmetic exception status bit in the user's TSA and returns CC1 set or reset according to the status value. The status bit is set whenever the user is in execution and an arithmetic exception trap occurs. The bit remains set until this service is requested, or the task terminates.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.TSTE.

Entry Conditions

Calling Sequence

M_TSTE

(or)

SVC 1,X'4D' (or) M_CALL H.REXS,23

Exit Conditions

Return Sequence

M.IPURTN

Registers

PSD CC1 contains the value of the arithmetic exception status bit

7.2.129 M_TSTS - Test User Status Word

The M_TSTS service returns the 32 bit user status word of any specified task in execution. The user status word resides in the CPU dispatch queue (DQE.USW) and is modified by the Set User Status Word (M_SETS) service. These two services treat the user status word as either a set of 32 flags or as a 32 bit counter. Bit 0 is used as a status flag.

The nonbase mode equivalent service is M.TSTS.

Entry Conditions

Calling Sequence

M_TSTS [TASK=]*task*

(or)

```
ZR   R6          } (or) LD R6,taskname
LW   R7,taskno   }
SVC  1,X'49' (or) M_CALL   H.REXS,8
```

task the address of a doubleword containing the name of a task or zero in word 1 and the task number in word 2. A task number must be used if the task is multicopied or shared. A task number of zero specifies the calling task.

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 bit 0 is set if the specified task was not found or the requesting task is not privileged and the owner name is restricted from access to tasks with a different owner name by the M.KEY file; otherwise, R7 returns the user-status word

7.2.130 M_TSTT - Test Timer Entry

The M_TSTT service returns to the caller the negative number of time units remaining until the specified timer entry time-out. If the timer has expired, the result returned is zero.

This service can be executed by the IPU.

The nonbase mode equivalent service is M.TSTT.

Entry Conditions

Calling Sequence

M_TSTT [TIMER=]*timer*

(or)

```
LW   R6,timer
SVC  1,X'46'  (or)  M_CALL   H.REXS,5
```

timer 2-character ASCII name of a timer, right-justified

Exit Conditions

Return Sequence

M.RTRN 7

Registers

R7 negative number of time units remaining until time-out or zero if the timer has expired or does not exist

7.2.131 M_TURNON - Activate Program at Given Time-of-Day

The M_TURNON service activates or resumes a specified task at a specified time and reactivates (resumes) it at specified intervals by creating a timer table entry using a specified timer ID. When a load module or executable image name is supplied as input, the operating system defaults to a search in the system directory only. For activations in other than the system directory, a pathname or RID vector must be supplied as input.

The nonbase mode equivalent service is M.TURNON.

Entry Conditions

Calling Sequence

M_TURNON [FNAME=]*filename* , [TIME=]*time* [, [RST=]*reset*],
[TIMID=]*timerid*

(or)

```
LD      R6,filename
LW      R4,time
LW      R5,reset
LW      R3,timerid
SVC     1,X'1E'    (or)    M_CALL    H.REXS,66
```

filename is either a left-justified blank-filled doubleword containing the 1- to 8-character ASCII name of the load module or executable image file (must be a system file),

(or)

R6 contains the pathname vector or RID vector which points to the task to be activated and R7 is zero

time is the time-of-day on the 24-hour clock when the task is activated. It is a word value with the following format:

Byte	Contents
0	binary hours
1	binary minutes
2	binary seconds
3	zero

reset is the time interval on the 24-hour clock to elapse before resetting the clock upon each time out. It has the same format as the time argument above. The task is reactivated at each time out. If a reset value is not specified, the comma denoting the field must still be specified and the task is activated only once.

timerid is a word variable containing the right-justified, zero-filled, 2-character ASCII name of the timer that will be created

Exit Conditions

Return Sequence

M.RTRN R3 nonzero

Error Condition

M.RTRN R3 zero if there are no timer entries available, the requested load module or executable image does not exist, attempting to create a duplicate timer ID, or invalid timer ID

7.2.132 M_TYPE - System Console Type

The M_TYPE service types a user specified message and optionally reads from the system console. Input message address is validated for the unprivileged task. Operation is wait I/O.

The maximum input or output is 80 characters. If no characters are specified, the maximum is used.

M_TYPE builds a type control parameter block (TCPB) which defines input and output buffer 24-bit addresses for console messages and reads.

The nonbase mode equivalent service is M.TYPE.

Entry Conditions

Calling Sequence

M_TYPE [OUTMES=]*addr1* , [OUTCNT=]*num1* [, [INMES=]*addr2* , [INCNT=]*num2*]

(or)

SVC 1,X'3F' (or) M_CALL H.IOCS,14

addr1	is the 24-bit address of the output message buffer
num1	is the transfer count for output. Up to 80 bytes can be transferred.
addr2	is the the 24-bit address of the input message buffer. If not specified, TCPB word 2 is zeroed. The first byte of this field contains the actual input quantity.
num2	is the transfer count for input. Up to 80 bytes can be transferred.

Registers

R1 contains address of the type control parameter block (TCPB)

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO15	A TASK HAS REQUESTED A TYPE OPERATION AND THE TYPE CONTROL PARAMETER BLOCK (TCPB) SPECIFIED INDICATES THAT AN OPERATION ASSOCIATED WITH THAT TCPB IS ALREADY IN PROGRESS

7.2.133 M_UNLOCK - Release Exclusive Resource Lock

The M_UNLOCK service is used with the Set Exclusive Resource Lock (M_LOCK) service. When called, the exclusive lock is released if the task has the resource allocated in a shareable mode; otherwise, the lock cannot be released until deallocation of the resource. Once the lock is released, other tasks can allocate the resource in a compatible access mode for the particular shared usage. However, another task is not able to exclusively lock the resource until this task, and all other sharing tasks, deallocate the resource.

The nonbase mode equivalent service is M.UNLOCK.

Entry Conditions

Calling Sequence

M_UNLOCK [ARGA=]*arga* [, [CNP=]*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr  (or)  ZR   R7
SVC  2,X'45'  (or)  M_CALL   H.REMM,24
```

arga is an address containing the allocation index obtained when the resource was assigned

(or)

an address containing the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC not assigned by this task
30	invalid allocation index
32	an exclusive resource lock was not owned by this task
33	resource is not allocated in a shareable mode by this task
46	unable to obtain resource descriptor lock (multiprocessor only)

Notes:

1. An exclusive resource lock can not be released by a task other than the owning task.
2. Any outstanding exclusive resource locks are released on task termination or on resource deallocation.

7.2.134 M_UNPRIVMODE - Change Task to Unprivileged Mode

The M_UNPRIVMODE service allows a task that was linked as privileged to operate as unprivileged. This causes the calling task's protection image to be loaded at every context switch. See the M_PRIVMODE service to reinstate privilege status.

The nonbase mode equivalent service is M.UPRIV.

Entry Conditions

Calling Sequence

M_UNPRIVMODE

(or)

SVC 2,X'58' (or) M_CALL H.REXS,79

Exit Conditions

Return Sequence

M.RTRN

7.2.135 M_UNSYNC - Release Synchronous Resource Lock

The M_UNSYNC service is used with the Set Synchronous Resource Lock (M_SETSYNC) service to perform gating on resources that have been allocated for explicit shared usage. When called, the synchronization lock is released, and all tasks waiting to own the lock are polled.

The nonbase mode equivalent service is M.UNSYNC.

Entry Conditions

Calling Sequence

M_UNSYNC [ARGA=]*arga* [, [CNP=]*cnpaddr*]

(or)

```
LW   R5,arga
LA   R7,cnpaddr  (or)  ZR   R7
SVC  2,X'47'  (or)  M_CALL   H.REMM,26
```

arga is an address containing the allocation index obtained when the resource was assigned

(or)

an address containing the address of a file control block (FCB) which contains an LFC in word 0

cnpaddr is the address of a caller notification packet (CNP) if notification is desired.

Applicable portions of the CNP for this function are abnormal return address and status field.

Exit Conditions

Return Sequence

(with CNP)	(without CNP)
M.RTRN	M.RTRN
(or)	(or)
M.RTNA (CC1 set)	M.RTRN R7 (CC1 set)

Registers

R7 return status if a CNP is not supplied; otherwise, unchanged

Status

CC1 set

Posted in R7 or the status field of the CNP:

Value	Description
29	specified LFC was not assigned by this task
30	invalid allocation index
32	synchronization lock was not set
46	unable to obtain resource descriptor lock (multiprocessor only)

Notes:

1. A synchronization lock may only be cleared by the task that set the lock.
2. A synchronization lock is automatically released when a task terminates or deallocates the resource.

7.2.136 M_UPSP - Upspace

The M_UPSP service is not applicable to blocked or SYC files. If BOT is present on multivolume magnetic tape, volume record (header) is written. If EOT is present on multivolume magnetic tape, an erase and write EOF is performed.

The nonbase mode equivalent service is M.UPSP.

Entry Conditions

Calling Sequence

M_UPSP [FCB=]*fcb*

(or)

```
LA   R1,fcb
SVC  1,X'10'  (or)  M_CALL  H.IOCS,20
```

fcb is the FCB address

Registers

R1 FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO06    INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED
        FILE ENCOUNTERED.  PROBABLE CAUSES: (1) FILE IS
        IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS
        DESTROYED, OR (3) TRANSFER ERROR DURING FILE
        INPUT.
IO09    ILLEGAL OPERATION ON THE SYC FILE
```

Output Messages

Mount/dismount messages if EOT on multivolume magnetic tape.

7.2.137 M_VADDR - Validate Address Range

The M_VADDR service verifies the specified logical address.

The nonbase mode equivalent service is M.VADDR.

Entry Conditions

Calling Sequence

M_VADDR [STARTADDR=]*addr* , [RANGE=]*num*

(or)

```
LW   R6, addr
LI   R7, num
SVC  2,X'59'   (or)   M_CALL   H.REXS,33
```

addr	is the logical starting address
num	is the number of bytes to validate

Registers

R6	contains *addr*
R7	contains *num*

Exit Conditions

Return Sequence

M.RTRN

Registers

CC2 set	address range crosses map block boundary
CC3 set	locations specified are protected
CC4 set	invalid address (not in caller's address space)
R0-R7	unchanged

7.2.138 M_WAIT - Wait I/O

The M_WAIT service provides return to the user when the I/O request associated with the specified FCB is complete. The task is suspended until I/O completes.

The nonbase mode equivalent service is M.WAIT.

Entry Conditions

Calling Sequence

M_WAIT [FCB=]*fcb*

(or)

```
LA   R1,fcb
SVC  1,X'3C'  (or)  M_CALL  H.IOCS,25
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
MS31    USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN
        END-ACTION ROUTINE
```

7.2.139 M_WRITE - Write Record

The M_WRITE service performs the following functions:

- prevents a write to a read-only file
- provides special random access handling for disk files
- blocks records for system and blocked files
- writes a volume record if BOT encountered on multivolume magnetic tape
- performs an erase and write EOF if EOT encountered on multivolume magnetic tape
- writes one record from the buffer pointed to by the TCW in the FCB

The nonbase mode equivalent service is M.WRIT.

Entry Conditions

Calling Sequence

M_WRITE [FCBADDR=]*addr*

(or)

LA R1, *addr*
SVC 1,X'32' (or) M_CALL H.IOCS,4

addr is the FCB address

Registers

R1 contains *addr*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO06 INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.

IO09 ILLEGAL OPERATION ON THE SYC FILE

IO38 WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE. A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF TASK HAS ONLY READ ACCESS TO UNIT.

RM02 ACCESS MODE NOT ALLOWED

Output Messages

Dismount/mount messages if EOT on multivolume magnetic tape.

7.2.140 M_WRITEEOF - Write EOF

The M_WRITEEOF service performs the following functions:

- prevents a write to a read-only file
- issues an end-of-file and purge if the file is a blocked file with an active blocking buffer
- writes a software EOF record (a 192-word record with X'0FE0FE0F' in its first word) immediately following the last record of an unblocked file created with EOFM=F
- writes a volume record if BOT is encountered on multivolume magnetic tape
- performs an erase and write EOF if EOT is encountered on multivolume magnetic tape

The nonbase mode equivalent service is M.WEOF.

Entry Conditions

Calling Sequence

M_WRITEEOF [FCBADDR=]*addr*

(or)

```
LA   R1, addr
SVC  1,X'38'   (or)   M_CALL  H.IOCS,5
```

addr is the FCB address

Registers

R1 contains *addr*

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09    ILLEGAL OPERATION ON THE SYC FILE
IO30    ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID
        ENCOUNTERED ON MAGNETIC TAPE
```

Output Messages

Dismount/mount messages if EOT encountered on multivolume magnetic tape.

7.2.141 M_XBRKR - Exit from Task Interrupt Level

The M_XBRKR service must be called at the conclusion of executing a task interrupt routine. It transfers control back to the point of interruption and resets the interrupt to the level established before the break or M_INT.

The nonbase mode equivalent service is M.XBRKR.

Entry Conditions

Calling Sequence

M_XBRKR

(or)

RETURN

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

RX34 TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO BREAK IS ACTIVE

7.2.142 M_XIEA - No-Wait I/O End-Action Return

The M_XIEA service is required to exit from a no-wait I/O end-action routine. Both normal and error end-action routines use this exit.

The nonbase mode equivalent service is M.XIEA.

Entry Conditions

Calling Sequence

M_XIEA

(or)

RETURN

Exit Conditions

Return Sequence

BL S.EXEC6 no-wait I/O postprocessing complete

7.2.143 M_XMEA - Exit from Message End-Action Routine

The M_XMEA service is called to exit the end-action routine associated with a no-wait message send request. For further description, see Chapter 2.

The nonbase mode equivalent service is M.XMEA.

Entry Conditions

Calling Sequence

M_XMEA

(or)

RETURN

Exit Conditions

Return Sequence

M.RTRN interrupts context at message interrupt or task base level

Abort Cases

RX99 TASK HAS MADE A MESSAGE END-ACTION ROUTINE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

7.2.144 M_XMSGR - Exit from Message Receiver

The M_XMSGR service must be called to exit the message receiver code of the calling task after the task has received a message from another task. For further description, see Chapter 2.

The nonbase mode equivalent service is M.XMSGR.

Entry Conditions

Calling Sequence

M_XMSGR [[RXB=]*rxbaddr*]

(or)

LA R1,*rxbaddr*
RETURN

rxbaddr is the logical address of the receiver exit block (RXB)

Exit Conditions

Return Sequence

M.RTRN interrupts context at task base level

Abort Cases

RX93 AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT

RX94 AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT

RX95 TASK HAS MADE A MESSAGE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

7.2.145 M_XREA - Exit from Run Request End-Action Routine

The M_XREA service is called to exit the end-action routine associated with having sent a no-wait run request.

The nonbase mode equivalent service is M.XREA.

Entry Conditions

Calling Sequence

M_XREA

(or)

RETURN

Exit Conditions

Return Sequence

M.RTRN interrupts context at message interrupt or task base level

Abort Cases

RX90 TASK HAS MADE A RUN REQUEST END-ACTION ROUTINE EXIT WHILE THE RUN REQUEST INTERRUPT WAS NOT ACTIVE

7.2.146 M_XRUNR - Exit Run Receiver

The M_XRUNR service is called to exit a task which was executing for a run request issued from another task.

The nonbase mode equivalent service is M.XRUNR.

Entry Conditions

Calling Sequence

M_XRUNR [[RXB=]*rxbaddr*]

(or)

LA R1,*rxbaddr*
RETURN

rxbaddr is the logical address of the receiver exit block (RXB). For further description, see Chapter 2.

Exit Conditions

Return Sequence

The run-receiver queue is examined and if not empty, the task is executed again on behalf of the next request. If the queue is empty, the exit options in the RXB are examined. If option byte is zero, the task is placed in a wait state, waiting for the next run request to be received. If option byte is nonzero, the task exits the system.

Note: If the task is re-executed, control is transferred to the instruction following the M_XRUNR call.

Abort Cases

RX96 AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT

RX97 AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT

RX98 TASK HAS MADE A RUN RECEIVER EXIT WHILE THE RUN RECEIVER INTERRUPT WAS NOT ACTIVE

7.2.147 M_XTIME - Task CPU Execution Time

The M_XTIME service returns to the caller the total accumulated processor execution time in microseconds since the intitiation of the task. If an IPU is present and IPU accounting is enabled, the time returned includes accumulated IPU execution time, if any. If the calling task is in the real time priority range and real time accounting is turned off, the returned time will be zero.

The nonbase mode equivalent service is M.XTIME.

Entry Conditions

Calling Sequence

M_XTIME

(or)

SVC 1,X'2D' (or) M_CALL H.REXS,65

Exit Conditions

Return Sequence

M.RTRN 6,7

Registers

R6,R7 CPU execution time in microseconds

7.3 Nonmacro-Callable System Services

7.3.1 Debug Link Service

The Debug Link service is used only by the interactive debugger to transfer control to the debugger. The debugger places this SVC trap in the user's task at the desired location.

Entry Conditions

Calling Sequence

SVC 1,X'66' (or) M_CALL H.REXS,42

Exit Conditions

Return Sequence

M.RTRN

7.3.2 Eject/Purge Routine

The Eject/Purge Routine service performs the following functions:

- if a file is blocked and output active, issues a purge and returns to the user
- writes a volume record if BOT is encountered on multivolume magnetic tape
- performs an erase and write EOF if EOT is encountered on multivolume magnetic tape
- eject is not applicable to SYC files

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'0D'   (or)   M_CALL H.IOCS,22
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09     ILLEGAL OPERATION ON THE SYC FILE
```

Output Messages

Mount/dismount messages if EOT encountered on multivolume magnetic tape.

7.3.3 Erase or Punch Trailer

The Erase or Punch Trailer service writes the volume record if BOT is encountered on multivolume magnetic tape, or performs an erase and write EOF if EOT is encountered on multivolume magnetic tape.

Erase or punch trailer is not applicable to blocked or SYC files.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'3E'   (or)   M_CALL   H.IOCS,21
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

```
IO09      ILLEGAL OPERATION ON THE SYC FILE
```

Output Messages

Mount/dismount messages if EOT encountered on multivolume magnetic tape.

7.3.4 Execute Channel Program

The Execute Channel Program service is available to privileged users and allows command and data chaining to General Purpose Multiplexer Controller (GPMC) and extended I/O devices only. Logical execute channel is available to both privileged and nonprivileged users. Physical execute channel is available only to privileged users.

Entry Conditions

Calling Sequence

```
LA   R1,fcb
SVC  1,X'25'   (or)   M_CALL H.IOCS,10
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

Abort Cases

IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO43	INPUT/OUTPUT CONTROL LIST (IOCL) OR DATA ADDRESS NOT IN CONTIGUOUS 'E' MEMORY (GPMC DEVICES ONLY)
IO50	AN UNPRIVILEGED USER ATTEMPTED TO EXECUTE A PHYSICAL CHANNEL PROGRAM
IO51	A 'TESTSTAR' COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO52	A LOGICAL CHANNEL WAS TOO LARGE TO BE MOVED TO MEMORY POOL
IO53	A 'TIC' COMMAND FOLLOWS A 'TIC' COMMAND IN A LOGICAL CHANNEL PROGRAM
IO54	A 'TIC' COMMAND ATTEMPTED TO TRANSFER TO AN ADDRESS WHICH IS NOT WORD BOUNDED
IO55	ILLEGAL ADDRESS IN LOGICAL IOCL. ADDRESS IS NOT IN USER'S LOGICAL ADDRESS SPACE
IO56	A READ-BACKWARD COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO57	ILLEGAL IOCL ADDRESS. IOCL MUST BE LOCATED IN THE FIRST 128K WORDS OF MEMORY

7.3.5 Get Extended Memory Array

The Get Extended Memory Array service requests an array of extended memory. If the request cannot be met, then all free memory, except one-eighth of the amount of physical memory, is allocated to the task and a count of maps allocated is returned. This service is intended for use by tasks that require the largest possible buffers without being placed on the MRQ for an extended period.

Entry Conditions

Calling Sequence

```
LW   R1,maps
SVC  2,X'7F'   (or)   M_CALL   H.MEMM,14
```

maps is the number of map blocks required

Exit Conditions

Return Sequence

M.RTRN

Registers

R2 number of map blocks allocated

R3 starting logical address of memory allocated or zero if an error occurred

R4 ending logical address of memory allocated or error code as follows:

Value	Description
1	CSECT overrun
2	request for more memory than physically exists
3	M_GETMEMBYTES service in use
4	unable to allocate logically contiguous memory

R5 number of map blocks, all classes, that are now free

7.3.6 Release FHD Port

The Release FHD Port service releases the fixed head disk port reserved by the Reserve FHD Port service. The release service is available only to privileged users and is only supported by the four megabyte fixed head disk.

Entry Conditions

Calling Sequence

```
LA      R1,fcb
SVC     1,X'27'   (or)   M_CALL   H.IOCS,27
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

7.3.7 Reserve FHD Port

The Reserve FHD Port service reserves a fixed head disk port. This service is available only to privileged tasks and is only supported by the four megabyte fixed head disk.

Entry Conditions

Calling Sequence

```
LA    R1,fcb
SVC   1,X'26'   (or)   M_CALL   H.IOCS,24
```

fcb is the FCB address

Exit Conditions

Return Sequence

M.RTRN

A MPX-32 Device Access

A.1 Description

Throughout the MPX-32 Reference Manual, the generic descriptor *devmnc* indicates that a device can be specified.

Under MPX-32, device addresses are specified using a combination of three levels of identification. They are device type, device channel/controller address, and device address/subaddress.

A device can be specified using the generic device type mnemonic only, which results in allocation of the first available device of the type requested. Device type mnemonics are listed in Table A-1.

A second method of device specification is achieved by using the generic device type mnemonic and specifying the channel/controller address. This results in allocation of the first available device of the type requested on the specified channel or controller.

The third method of device selection requires specification of the device type mnemonic, channel/controller, and device address/subaddress. This method allows specification of a particular device.

Table A-1
Device Type Mnemonics and Codes

Device Type Code	Device Type Mnemonic	Device Description
00	CT	Operator console (not assignable)
01	DC	Any disk unit except memory disk
02	DM	Any moving head or memory disk
03	DF	Any fixed head disk
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 disk
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 disk except memory disk

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

A.2 Special Device Specifications and Handling

A.2.1 Magnetic Tape/Floppy Disk

For magnetic tape and floppy disks, unblocking, density, a reel identifier, and multivolume number (magnetic tape only) can be included in the device specification.

Syntax

$ASSIGN *lfc* **TO DEV=***devmnc* [**BLO**CKED=**{Y|N}**]
[**DEN**SITY=**{N|P|G|800|1600|6250}**] [**ID=***id*] [**MUL**TIVOL=*number*]

lfc is a 1- to 3-character logical file code

DEV=*devmnc*
devmnc is the device specification of a configured peripheral device (see the Description section)

[**BLO**CKED=**{Y|N}**]
if Y is specified, medium is blocked. If N is specified, medium is not blocked. If not specified the default is blocked.

[**DEN**SITY=**{N|P|G|800|1600|6250}**]
specifies density of high speed XIO tape. If not specified, the default is 6250 bpi. Values are as follows:

Value	Description
N or 800	indicates 800 bpi nonreturn to zero inverted (NRZI).
P or 1600	indicates 1600 bpi phase encoded (PE).
G or 6250	indicates 6250 bpi group coded recording (GCR). This is the default.

[**ID=***id*] *id* specifies a 1- to 4-character identifier for the reel. If not specified, the default is SCRA (scratch).

[**MUL**TIVOL=*number*]
number is a volume number. If multivolume tape, *number* must be specified. If not specified, the default is not multivolume (0). This option is not valid for use with floppy disks.

When the task that has an assignment to tape is activated, a mount message indicates the name of the task and other information on the system console:

```
MOUNT reel VOL volume ON devmnc
TASK taskname,taskno REPLY R,H,A, OR DEVICE:
jobno
```

reel specifies a 1- to 4-character identifier for the reel. If not specified, the default is SCRA (Scratch).

volume identifies the volume number to mount if multivolume tape

devmnc is the device mnemonic for the tape unit selected in response to the assignment. If a specific channel and subaddress are supplied in the assignment, the specific tape drive is selected and named in the message; otherwise, a unit is selected by the system and its complete address is named in the message.

jobno identifies the job by job number if the task is part of a batch job

taskname is the name of the task to which the tape is assigned

taskno is the task number assigned to the task by the system

`R,H,A, OR DEVICE`
the device listed in the message can be allocated and the task resumed (`R`), a different device can be selected (`DEVICE`), the task can be aborted (`A`), or the task can be held with the specified device deallocated (`H`). If an R response is given and a high speed XIO tape drive is being used, its density can be changed when the software select feature is enabled on the tape unit front panel. If specified, it overrides any specification made at assignment. Example usage: `RN`, `R1600`, etc.

Note: Do not insert blanks or commas.

Response:

To indicate the drive specified in the mount message is ready and proceed with the task, mount the tape on the drive and type `R` (resume), optionally followed by a density specification if the drive is a high speed XIO tape unit. To abort the task, type `A` (abort). To hold the task and deallocate the specified device, type `H` (hold). The task can be resumed by the OPCOM CONTINUE directive; at which time, a tape drive is selected by the system and the mount message redisplayed.

To select a tape drive other than the drive specified in the message, enter the mnemonic of the drive to be used. Any of the three levels of device identification can be used. The mount message is reissued. Mount the tape and type `R` if satisfactory, or if not satisfactory, abort, override, or hold as described.

Examples of the three methods of device specification follow:

Type 1 - Generic Device Class

```
$ASSIGN OUT TO DEV=M9 MUL=1 ID=MVOL
```

In this example, the device assigned to logical file code (LFC) `OUT` is any `9`-track tape unit on any channel. The multivolume reel number is `1`. The reel identifier is `MVOL` and the tape is blocked.

Type 2 - Generic Device Class and Channel/Controller

```
$ASSIGN OUT TO DEV=M910 ID=MVOL BLO=N
```

In this example, the device assigned to logical file code (LFC) `OUT` is the first available 9-track tape unit on channel `10`. The specification is invalid if a 9-track tape unit does not exist on the channel. The reel identifier is `MVOL`. This is not a multivolume tape and is unblocked.

Type 3 - Specific Device Request

```
$ASSIGN OUT TO DEV=M91001
```

In this example, the device assigned to logical file code (LFC) `OUT` is the 9-track tape unit `01` on channel `10`. The specification is invalid if unit 01 on channel 10 is not a 9-track tape. The tape reel identifier is SCRA. The tape is blocked and is not multivolume.

A.2.2 Temporary Disk Space

For a temporary disk file the following can be specified: size, blocking, printing or punching, and access.

Syntax

$ASSIGN *lfc* **TO TEM**P[=(*volname*)] [**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])] [**BLO**CKED={**Y** | **N**}] [**PRI**NT | **PUN**CH] [**SIZ**E=*blocks*]

lfc is a 1- to 3-character logical file code

TEMP[=(*volname*)]
volname is the 1- to 16-character volume name where temporary space is allocated. If not specified, the default is the current working volume or any public volume.

[**ACC**ESS=([**R**EAD] [**W**RITE] [**M**ODIFY] [**U**PDATE] [**A**PPEND])]
specifies the types of access for the file. If not specified, the default is the access specified at file creation.

[**BLO**CKED={**Y** | **N**}]
if Y is specified, the file is blocked. If N is specified, the file is unblocked. If not specified, the default is blocked.

[**PRI**NT | **PUN**CH]
indicates the file is to be printed (PRINT) or punched (PUNCH) after deassignment

[**SIZ**E=*blocks*]
blocks is the number of 192-word blocks required. If not specified, the default is 16 blocks.

Examples

In the following example, the device assigned to logical file code (LFC) OUT is the current working volume or any public volume and the file prints to the SLO device after deassignment.

```
AS OUT TO TEM PRI
```

The following example designates the system volume as the device for the temporary blocked file.

```
AS OUT TO TEMP=(SYSTEM) BLO=Y
```

A.3 GPMC Devices

GPMC/GPDC device specifications follow the general structure just described. The terminal at subaddress 04 on GPMC 01 whose channel address is 20 would be identified as follows:

```
$AS DEV TO DEV=TY2004
```

A.4 Null Device

A special device type, NU, is available for null device specifications. Files accessed using this device type generate an end-of-file (EOF) when a read is attempted and normal completion when a write is attempted.

A.5 System Console

Logical file codes are assigned to the system console by using the device type CT.

A.6 Special System Files

There are four special mnemonics provided for access to special system files: SLO, SBO, SGO and SYC. These are assigned with the $ASSIGN statement, as in:

```
$ASSIGN OUT TO SLO
```

For nonbatch tasks, SLO and SBO files are allocated dynamically by the system and used to disk buffer output to a device selected automatically. For batch tasks, use of SLO and SBO files is identical, except that automatic selection of a device can be overridden by assigning a specific file or device.

A.7 Samples

A description of device selection possibilities is constructed as follows:

Disk

DC	Any disk except memory disk
DM	Any moving head or memory disk
DM08	Any moving head disk on channel 08
DM0801	Moving head disk 01 on channel 08
DM0002	Memory disk 02 on channel 00
DF	Any fixed head disk
DF04	Any fixed head disk on channel 04
DF0401	Fixed head disk 01 on channel 04

Tape

MT	Any magnetic tape
M9	Any 9-track magnetic tape
M910	Any 9-track magnetic tape on channel 10
M91002	9-track magnetic tape 02 on channel 10

Card Equipment

CR	Any card reader
CR78	Any card reader on channel 78
CR7800	Card reader 00 on channel 78

Line Printer

LP	Any line printer
LP7A	Any line printer on channel 7A
LP7A00	Line printer 00 on channel 7A
LP7EA0	Serial printer A0 on ACM channel 7E

B System Services Cross-Reference

B.1 Macro Name Listing

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.ACTV	Activate Task	1,X'52'	H.REXS,15	6.2
M_ACTV	Activate Task	1,X'52'	H.REXS,15	7.2
M.ADRS	Memory Address Inquiry	1,X'44'	H.REXS,3	6.2
M_ADRS	Memory Address Inquiry	1,X'44'	H.REXS,3	7.2
M_ADVANCE	Advance Record	1,X'33'	H.IOCS,7	7.2
	Advance File	1,X'34'	H.IOCS,8	7.2
M.ALOC	Allocate File or Peripheral Device	1,X'40'	H.MONS,21	6.4
M.ANYW	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	1,X'7C'	H.REXS,37	6.2
M_ANYWAIT	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	1,X'7C'	H.REXS,37	7.2
M_ASSIGN	Assign and Allocate Resource	2,X'52'	H.REXS,21	7.2
M.ASSN	Assign and Allocate Resource	2,X'52'	H.REXS,21	6.2
M.ASYNCH	Set Asynchronous Task Interrupt	1,X'1C'	H.REXS,68	6.2
M_ASYNCH	Set Asynchronous Task Interrupt	1,X'1C'	H.REXS,68	7.2
M_AWAITACTION	End Action Wait	1,X'1D'	H.EXEC,40	7.2
M.BACK	Backspace Record	1,X'35'	H.IOCS,9	6.2
	Backspace File	1,X'36'	H.IOCS,19	6.2
M_BACKSPACE	Backspace Record	1,X'35'	H.IOCS,9	7.2
	Backspace File	1,X'36'	H.IOCS,19	7.2
M.BATCH	Batch Job Entry	2,X'55'	H.REXS,27	6.2
M_BATCH	Batch Job Entry	2,X'55'	H.REXS,27	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.BBTIM	Acquire Current Date/Time in Byte Binary Format	2,X'50'	H.REXS,74	6.2
M_BBTIM	Acquire Current Date/Time in Byte Binary Format	2,X'50'	H.REXS,74	7.2
M.BORT	Abort Specified Task	1,X'56'	H.REXS,19	6.2
	Abort Self	1,X'57'	H.REXS,20	6.2
	Abort With Extended Message	1,X'62'	H.REXS,28	6.2
M_BORT	Abort Specified Task	1,X'56'	H.REXS,19	7.2
	Abort Self	1,X'57'	H.REXS,20	7.2
	Abort With Extended Message	1,X'62'	H.REXS,28	7.2
M.BRK	Break/Task Interrupt Link/Unlink	1,X'6E'	H.REXS,46	6.2
M_BRK	Break/Task Interrupt Link/Unlink	1,X'6E'	H.REXS,46	7.2
M.BRKXIT	Exit From Task Interrupt Level	1,X'70'	H.REXS,48	6.2
M_BRKXIT	Exit From Task Interrupt Level	N/A	N/A	7.2
M.BTIM	Acquire Current Date/Time in Binary Format	2,X'50'	H.REXS,74	6.2
M_BTIM	Acquire Current Date/Time in Binary Format	2,X'50'	H.REXS,74	7.2
M.CDJS	Submit Job from Disc File	1,X'61'	H.MONS,27	6.4
M_CHANPROGFCB	Execute Channel Program File Control Block	N/A	N/A	7.2
M.CLOSER	Close Resource	2,X'43'	H.REMM,22	6.2
M_CLOSER	Close Resource	2,X'43'	H.REMM,22	7.2
M.CLSE	Close File	1,X'39'	H.IOCS,23	6.2
M_CLSE	Close File	1,X'39'	H.IOCS,23	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.CMD	Get Command Line	2,X'61'	H.REXS,88	6.2
M_CMD	Get Command Line	2,X'61'	H.REXS,88	7.2
M.CONABB	Convert ASCII Date/Time to Byte Binary Format	2,X'51'	H.REXS,75	6.2
M_CONABB	Convert ASCII Date/Time to Byte Binary Format	2,X'51'	H.REXS,75	7.2
M.CONADB	Convert ASCII Decimal to Binary	1,X'28'	H.TSM,7	6.2
M_CONADB	Convert ASCII Decimal to Binary	1,X'28'	H.TSM,7	7.2
M.CONAHB	Convert ASCII Hex to Binary	1,X'29'	H.TSM,8	6.2
M_CONAHB	Convert ASCII Hex to Binary	1,X'29'	H.TSM,8	7.2
M.CONASB	Convert ASCII Date/Time to Standard Binary	2,X'51'	H.REXS,75	6.2
M_CONASB	Convert ASCII Date/Time to Standard Binary	2,X'51'	H.REXS,75	7.2
M.CONBAD	Convert Binary to ASCII Decimal	1,X'2A'	H.TSM,9	6.2
M_CONBAD	Convert Binary to ASCII Decimal	1,X'2A'	H.TSM,9	7.2
M.CONBAF	Convert Binary Date/Time to ASCII Format	2,X'51'	H.REXS,75	6.2
M_CONBAF	Convert Binary Date/Time to ASCII Format	2,X'51'	H.REXS,75	7.2
M.CONBAH	Convert Binary to ASCII Hex	1,X'2B'	H.TSM,10	6.2
M_CONBAH	Convert Binary to ASCII Hex	1,X'2B'	H.TSM,10	7.2
M.CONBBA	Convert Byte Binary Date/Time to ASCII	2,X'51'	H.REXS,75	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_CONBBA	Convert Byte Binary Date/Time to ASCII	2,X'51'	H.REXS,75	7.2
M.CONBBY	Convert Binary Date/Time to Byte Binary	2,X'51'	H.REXS,75	6.2
M_CONBBY	Convert Binary Date/Time to Byte Binary	2,X'51'	H.REXS,75	7.2
M.CONBYB	Convert Byte Binary Date/Time to Binary	2,X'51'	H.REXS,75	6.2
M_CONBYB	Convert Byte Binary Date/Time to Binary	2,X'51'	H.REXS,75	7.2
M.CONN	Connect Task to Interrupt	1,X'4B'	H.REXS,10	6.2
M_CONN	Connect Task to Interrupt	1,X'4B'	H.REXS,10	7.2
M_CONSTRUCTPATH	Reconstruct Pathname	2,X'2F'	H.VOMM,16	7.2
M_CONVERTTIME	Convert Time	2,X'51'	H.REXS,75	7.2
M.CPERM	Create Permanent File	2,X'20'	H.VOMM,1	6.2
M.CREATE	Create Permanent File	1,X'75'	H.FISE,12	6.4
M_CREATEFCB	Create File Control Block	N/A	N/A	7.2
M_CREATEP	Create Permanent File	2,X'20'	H.VOMM,1	7.2
M_CREATET	Create Temporary File	2,X'21'	H.VOMM,2	7.2
M.CTIM	Convert System Date/Time Format	2,X'51'	H.REXS,75	6.2
M_CTIM	Convert System Date/Time Format	2,X'51'	H.REXS,75	7.2
M.CWAT	System Console Wait	1,X'3D'	H.IOCS,26	6.2
M_CWAT	System Console Wait	1,X'3D'	H.IOCS,26	7.2
M.DALC	Deallocate File or Peripheral Device	1,X'41'	H.MONS,22	6.4
M.DASN	Deassign and Deallocate Resource	2,X'53'	H.REXS,22	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.DATE	Date and Time Inquiry	1,X'15'	H.REXS,70	6.2
M_DATE	Date and Time Inquiry	1,X'15'	H.REXS,70	7.2
M_DEASSIGN	Deassign and Deallocate Resource	2,X'53'	H.REXS,22	7.2
M.DEBUG	Load and Execute Interactive Debugger	1,X'63'	H.REXS,29	6.2
M_DEBUG	Load and Execute Interactive Debugger	1,X'63'	H.REXS,29	7.2
M.DEFT	Change Defaults	2,X'27'	H.VOMM,8	6.2
M_DEFT	Change Defaults	2,X'27'	H.VOMM,8	7.2
M.DELETE	Delete Permanent File or Non-SYSGEN Memory Partition	1,X'77'	H.FISE,14	6.4
M_DELETER	Delete Resource	2,X'24'	H.VOMM,5	7.2
M.DELR	Delete Resource	2,X'24'	H.VOMM,5	6.2
M.DELTSK	Delete Task	1,X'5A'	H.REXS,31	6.2
M_DELTSK	Delete Task	1,X'5A'	H.REXS,31	7.2
M.DEVID	Get Device Mnemonic or Type Code	1,X'14'	H.REXS,71	6.2
M_DEVID	Get Device Mnemonic or Type Code	1,X'14'	H.REXS,71	7.2
M.DFCB	Create File Control Block	N/A	N/A	5.9.1
M.DIR	Create Directory	2,X'23'	H.VOMM,4	6.2
M_DIR	Create Directory	2,X'23'	H.VOMM,4	7.2
M.DISCON	Disconnect Task from Interrupt	1,X'5D'	H.REXS,38	6.2
M_DISCON	Disconnect Task from Interrupt	1,X'5D'	H.REXS,38	7.2
M_DISMOUNT	Dismount Volume	2,X'4A'	H.REMM,19	7.2
M.DLTT	Delete Timer Entry	1,X'47'	H.REXS,6	6.2
M_DLTT	Delete Timer Entry	1,X'47'	H.REXS,6	7.2
M.DMOUNT	Dismount Volume	2,X'4A'	H.REMM,19	6.2

Macro Name Listing

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.DSMI	Disable Message Task Interrupt	1,X'2E'	H.REXS,57	6.2
M_DSMI	Disable Message Task Interrupt	1,X'2E'	H.REXS,57	7.2
M.DSUB	Disable User Break Interrupt	1,X'12'	H.REXS,73	6.2
M_DSUB	Disable User Break Interrupt	1,X'12'	H.REXS,73	7.2
M.DUMP	Memory Dump Request	1,X'4F'	H.REXS,12	6.2
M_DUMP	Memory Dump Request	1,X'4F'	H.REXS,12	7.2
M.EAWAIT	End Action Wait	1,X'1D'	H.EXEC,40	6.2
M.ENMI	Enable Message Task Interrupt	1,X'2F'	H.REXS,58	6.2
M_ENMI	Enable Message Task Interrupt	1,X'2F'	H.REXS,58	7.2
M.ENUB	Enable User Break Interrupt	1,X'13'	H.REXS,72	6.2
M_ENUB	Enable User Break Interrupt	1,X'13'	H.REXS,72	7.2
M.ENVRMT	Get Task Environment	2,X'5E'	H.REXS,85	6.2
M_ENVRMT	Get Task Environment	2,X'5E'	H.REXS,85	7.2
M.EXCL	Free Shared Memory	1,X'79'	H.ALOC,14	6.4
M.EXCLUDE	Exclude Memory Partition	2,X'41'	H.REMM,14	6.2
M_EXCLUDE	Exclude Shared Image	2,X'41'	H.REMM,14	7.2
M.EXIT	Terminate Task Execution	1,X'55'	H.REXS,18	6.2
M_EXIT	Terminate Task Execution	1,X'55'	H.REXS,18	7.2
M.EXTD	Extend File	2,X'25'	H.VOMM,6	6.2
M_EXTENDFILE	Extend File	2,X'25'	H.VOMM,6	7.2
M_EXTSTS	Exit With Status	2,X'5F'	H.REXS,86	7.2
M.FADD	Permanent File Address Inquiry	1,X'43'	H.MONS,2	6.4

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.FD	Free Dynamic Extended Indexed Data Space	1,X'6A'	H.REMM,9	6.2
M.FE	Free Dynamic Task Execution Space	1,X'68'	H.REMM,11	6.2
M.FILE	Open File	1,X'30'	H.IOCS,1	6.4
M_FREEMEMBYTES	Free Memory in Byte Increments	2,X'4C'	H.REMM,29	7.2
M.FSLR	Release Synchronization File Lock	1,X'24'	H.FISE,25	6.4
M.FSLS	Set Synchronization File Lock	1,X'23'	H.FISE,24	6.4
M.FWRD	Advance Record	1,X'33'	H.IOCS,7	6.2
	Advance File	1,X'34'	H.IOCS,8	6.2
M.FXLR	Release Exclusive File Lock	1,X'22'	H.FISE,23	6.4
M.FXLS	Set Exclusive File Lock	1,X'21'	H.FISE,22	6.4
M.GADRL	Get Address Limits	1,X'65'	H.REXS,41	6.2
M.GADRL2	Get Address Limits	2,X'7B'	H.REXS,80	6.2
M.GD	Get Dynamic Extended Data Space	1,X'69'	H.REMM,8	6.2
M.GDD	Get Dynamic Extended Discontiguous Data Space	2,X'7C'	H.MEMM,9	6.2
M.GE	Get Dynamic Task Execution Space	1,X'67'	H.REMM,10	6.2
M_GETCTX	Get User Context	2,X'70'	H.EXEC,41	7.2
M.GETDEF	Get Terminal Function Definition	2,X'7A'	H.TSM,15	6.2
M_GETDEF	Get Terminal Function Definition	2,X'7A'	H.TSM,15	7.2
M_GETMEMBYTES	Get Memory in Byte Increments	2,X'4B'	H.REMM,28	7.2
M_GETTIME	Get Current Date and Time	2,X'50'	H.REXS,74	7.2
M.GMSGP	Get Message Parameters	1,X'7A'	H.REXS,35	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_GMSGP	Get Message Parameters	1,X'7A'	H.REXS,35	7.2
M.GRUNP	Get Run Parameters	1,X'7B'	H.REXS,36	6.2
M_GRUNP	Get Run Parameters	1,X'7B'	H.REXS,36	7.2
M.GTIM	Acquire System Date/Time in Any Format	2,X'50'	H.REXS,74	6.2
M_GTIM	Acquire System Date/Time in Any Format	2,X'50'	H.REXS,74	7.2
M.GTSAD	Get TSA Start Address	2,X'7D'	H.REXS,91	6.2
M_GTSAD	Get TSA Start Address	2,X'7D'	H.REXS,91	7.2
M.HOLD	Program Hold Request	1,X'58'	H.REXS,25	6.2
M_HOLD	Program Hold Request	1,X'58'	H.REXS,25	7.2
M.ID	Get Task Number	1,X'64'	H.REXS,32	6.2
M_ID	Get Task Number	1,X'64'	H.REXS,32	7.2
M.INCL	Get Shared Memory	1,X'72'	H.ALOC,13	6.4
M.INCLUDE	Include Memory Partition	2,X'40'	H.REMM,12	6.2
M_INCLUDE	Include Shared Image	2,X'40'	H.REMM,12	7.2
M_INQUIRER	Resource Inquiry	2,X'48'	H.REMM,27	7.2
M.INQUIRY	Resource Inquiry	2,X'48'	H.REMM,27	6.2
M.INT	Activate Task Interrupt	1,X'6F'	H.REXS,47	6.2
M_INT	Activate Task Interrupt	1,X'6F'	H.REXS,47	7.2
M.IPUBS	Set IPU Bias	2,X'5B'	H.REXS,82	6.2
M_IPUBS	Set IPU Bias	2,X'5B'	H.REXS,82	7.2
M_LIMITS	Get Base Mode Task Address Limits	2,X'5D'	H.REXS,84	7.2
M.LOC	Read Descriptor	2,X'2C'	H.VOMM,13	6.2
M.LOCK	Set Exclusive Resource Lock	2,X'44'	H.REMM,23	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_LOCK	Set Exclusive Resource Lock	2,X'44'	H.REMM,23	7.2
M.LOG	Permanent File Log	1,X'73'	H.MONS,33	6.4
M.LOGR	Log Resource or Directory	2,X'29'	H.VOMM,10	6.2
M_LOGR	Log Resource or Directory	2,X'29'	H.VOMM,10	7.2
M.MEM	Create Memory Partition	2,X'22'	H.VOMM,3	6.2
M_MEM	Create Memory Partition	2,X'22'	H.VOMM,3	7.2
M.MEMB	Get Memory in Byte Increments	2,X'4B'	H.REMM,28	6.2
M.MEMFRE	Free Memory in Byte Increments	2,X'4C'	H.REMM,29	6.2
M.MOD	Modify Descriptor	2,X'2A'	H.VOMM,11	6.2
M_MOD	Modify Descriptor	2,X'2A'	H.VOMM,11	7.2
M.MODU	Modify Descriptor User Area	2,X'31'	H.VOMM,26	6.2
M_MODU	Modify Descriptor User Area	2,X'31'	H.VOMM,26	7.2
M.MOUNT	Mount Volume	2,X'49'	H.REMM,17	6.2
M_MOUNT	Mount Volume	2,X'49'	H.REMM,17	7.2
M.MOVE	Move Data to User Address	2,X'62'	H.REXS,89	6.2
M_MOVE	Move Data to User Address	2,X'62'	H.REXS,89	7.2
M.MYID	Get Task Number	1,X'64'	H.REXS,32	6.2
M_MYID	Get Task Number	1,X'64'	H.REXS,32	7.2
M.NEWRRS	Reformat RRS Entry	2,X'54'	H.REXS,76	6.2
M.OLAY	Load Overlay Segment Load and Execute Overlay	1,X'50' 1,X'51'	H.REXS,13 H.REXS,14	6.2 6.2
M.OPENR	Open Resource	2,X'42'	H.REMM,21	6.2
M_OPENR	Open Resource	2,X'42'	H.REMM,21	7.2
M_OPTIONDWORD	Task Option Doubleword Inquiry	2,X'C0'	H.REXS,95	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_OPTIONWORD	Task Option Word Inquiry	1,X'4C'	H.REXS,24	7.2
M.OSREAD	Physical Memory Read	2,X'7E'	H.REXS,93	6.2
M_OSREAD	Physical Memory Read	2,X'7E'	H.REXS,93	7.2
M.OSWRIT	Physical Memory Write	2,X'AF'	H.REXS,94	6.2
M_OSWRIT	Physical Memory Write	2,X'AF'	H.REXS,94	7.2
M.PDEV	Physical Device Inquiry	1,X'42'	H.MONS,1	6.4
M.PERM	Change Temporary File to Permanent	1,X'76'	H.FISE,13	6.4
M.PGOD	Task Option Doubleword Inquiry	2,X'C0'	H.REXS,95	6.2
M.PGOW	Task Option Word Inquiry	1,X'4C'	H.REXS,24	6.2
M.PNAM	Reconstruct Pathname	2,X'2F'	H.VOMM,16	6.2
M.PNAMB	Convert Pathname to Pathname Block	2,X'2E'	H.VOMM,15	6.2
M_PNAMB	Convert Pathname to Pathname Block	2,X'2E'	H.VOMM,15	7.2
M.PRIL	Change Priority Level	1,X'4A'	H.REXS,9	6.2
M_PRIL	Change Priority Level	1,X'4A'	H.REXS,9	7.2
M.PRIV	Reinstate Privilege Mode to Privilege Task	2,X'57'	H.REXS,78	6.2
M_PRIVMODE	Reinstate Privilege Mode to Privilege Task	2,X'57'	H.REXS,78	7.2
M.PTSK	Parameter Task Activation	1,X'5F'	H.REXS,40	6.2
M_PTSK	Parameter Task Activation	1,X'5F'	H.REXS,40	7.2
M_PUTCTX	Put User Context	2,X'71'	H.EXEC,42	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.QATIM	Acquire Current Date/Time in ASCII Format	2,X'50'	H.REXS,74	6.2
M_QATIM	Acquire Current Date/Time in ASCII Format	2,X'50'	H.REXS,74	7.2
M.RADDR	Get Real Physical Address	1,X'0E'	H.REXS,90	6.2
M_RADDR	Get Real Physical Address	1,X'0E'	H.REXS,90	7.2
M.RCVR	Receive Message Link Address	1,X'6B'	H.REXS,43	6.2
M_RCVR	Receive Message Link Address	1,X'6B'	H.REXS,43	7.2
M.READ	Read Record	1,X'31'	H.IOCS,3	6.2
M_READ	Read Record	1,X'31'	H.IOCS,3	7.2
M_READD	Read Descriptor	2,X'2C'	H.VOMM,13	7.2
M.RELP	Release Dual-ported Disc/Set Dual-channel ACM Mode	1,X'27'	H.IOCS,27	6.2
M_RELP	Release Dual-ported Disc/Set Dual-channel ACM Mode	1,X'27'	H.IOCS,27	7.2
M.RENAM	Rename File	2,X'2D'	H.VOMM,14	6.2
M_RENAME	Rename File	2,X'2D'	H.VOMM,14	7.2
M.REPLAC	Replace Permanent File	2,X'30'	H.VOMM,23	6.2
M_REPLACE	Replace Permanent File	2,X'30'	H.VOMM,23	7.2
M.RESP	Reserve Dual-ported Disc/Set Single-channel ACM Mode	1,X'26'	H.IOCS,24	6.2
M_RESP	Reserve Dual-ported Disc/Set Single-channel ACM Mode	1,X'26'	H.IOCS,24	7.2
M_REWIND	Rewind File	1,X'37'	H.IOCS,2	7.2
M.REWRIT	Rewrite Descriptor	2,X'2B'	H.VOMM,12	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_REWRIT	Rewrite Descriptor	2,X'2B'	H.VOMM,12	7.2
M.REWRTU	Rewrite Descriptor User Area	2,X'32'	H.VOMM,27	6.2
M_REWRTU	Rewrite Descriptor User Area	2,X'32'	H.VOMM,27	7.2
M.ROPL	Reset Option Lower	2,X'78'	H.TSM,14	6.2
M_ROPL	Reset Option Lower	2,X'78'	H.TSM,14	7.2
M.RRES	Release Channel Reservation	1,X'3B'	H.IOCS,13	6.2
M_RRES	Release Channel Reservation	1,X'3B'	H.IOCS,13	7.2
M.RSML	Resourcemark Lock	1,X'19'	H.REXS,62	6.2
M_RSML	Resourcemark Lock	1,X'19'	H.REXS,62	7.2
M.RSMU	Resourcemark Unlock	1,X'1A'	H.REXS,63	6.2
M_RSMU	Resourcemark Unlock	1,X'1A'	H.REXS,63	7.2
M.RSRV	Reserve Channel	1,X'3A'	H.IOCS,12	6.2
M_RSRV	Reserve Channel	1,X'3A'	H.IOCS,12	7.2
M.RWND	Rewind File	1,X'37'	H.IOCS,2	6.2
M_SETERA	Set Exception Return Address	2,X'79'	H.REXS,81	7.2
M_SETEXA	Set Exception Handler	2,X'5C'	H.REXS,83	7.2
M.SETS	Set User Status Word	1,X'48'	H.REXS,7	6.2
M_SETS	Set User Status Word	1,X'48'	H.REXS,7	7.2
M.SETSYNC	Set Synchronous Resource Lock	2,X'46'	H.REMM,25	6.2
M_SETSYNC	Set Synchronous Resource Lock	2,X'46'	H.REMM,25	7.2
M.SETT	Create Timer Entry	1,X'45'	H.REXS,4	6.2
M_SETT	Create Timer Entry	1,X'45'	H.REXS,4	7.2
M.SHARE	Share Memory with Another Task	1,X'71'	H.ALOC,12	6.4
M.SMSGR	Send Message to Specified Task	1,X'6C'	H.REXS,44	6.2
M_SMSGR	Send Message to Specified Task	1,X'6C'	H.REXS,44	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.SMULK	Unlock and Dequeue Shared Memory	1,X'1F'	H.ALOC,19	6.4
M.SOPL	Set Option Lower	2,X'77'	H.TSM,13	6.2
M_SOPL	Set Option Lower	2,X'77'	H.TSM,13	7.2
M.SRUNR	Send Run Request to Specified Task	1,X'6D'	H.REXS,45	6.2
M_SRUNR	Send Run Request to Specified Task	1,X'6D'	H.REXS,45	7.2
M.SUAR	Set User Abort Receiver Address	1,X'60'	H.REXS,26	6.2
M_SUAR	Set User Abort Receiver Address	1,X'60'	H.REXS,26	7.2
M.SUME	Resume Task Execution	1,X'53'	H.REXS,16	6.2
M_SUME	Resume Task Execution	1,X'53'	H.REXS,16	7.2
M.SURE	Suspend/Resume	5,X'00'	N/A	6.2
M_SURE	Suspend/Resume	5,X'00'	N/A	7.2
M.SUSP	Suspend Task Execution	1,X'54'	H.REXS,17	6.2
M_SUSP	Suspend Task Execution	1,X'54'	H.REXS,17	7.2
M.SYNCH	Set Synchronous Task Interrupt	1,X'1B'	H.REXS,67	6.2
M_SYNCH	Set Synchronous Task Interrupt	1,X'1B'	H.REXS,67	7.2
M.TBRKON	Trap On-line User's Task	1,X'5C'	H.TSM,6	6.2
M_TBRKON	Trap On-line User's Task	1,X'5C'	H.TSM,6	7.2
M.TDAY	Time-of-Day Inquiry	1,X'4E'	H.REXS,11	6.2
M_TDAY	Time-of-Day Inquiry	1,X'4E'	H.REXS,11	7.2
M.TEMP	Create Temporary File	2,X'21'	H.VOMM,2	6.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.TEMPER	Change Temporary File to Permanent File	2,X'28'	H.VOMM,9	6.2
M_TEMPFILETOPERM	Change Temporary File to Permanent File	2,X'28'	H.VOMM,9	7.2
M.TRNC	Truncate File	2,X'26'	H.VOMM,7	6.2
M_TRUNCATE	Truncate File	2,X'26'	H.VOMM,7	7.2
M.TSCAN	Scan Terminal Input Buffer	1,X'5B'	H.TSM,2	6.2
M_TSCAN	Scan Terminal Input Buffer	1,X'5B'	H.TSM,2	7.2
M.TSMPC	TSM Procedure Call	2,X'AE'	H.TSM,17	6.2
M_TSMPC	TSM Procedure Call	2,X'AE'	H.TSM,17	7.2
M.TSTE	Arithmetic Exception Inquiry	1,X'4D'	H.REXS,23	6.2
M_TSTE	Arithmetic Exception Inquiry	1,X'4D'	H.REXS,23	7.2
M.TSTS	Test User Status Word	1,X'49'	H.REXS,8	6.2
M_TSTS	Test User Status Word	1,X'49'	H.REXS,8	7.2
M.TSTT	Test Timer Entry	1,X'46'	H.REXS,5	6.2
M_TSTT	Test Timer Entry	1,X'46'	H.REXS,5	7.2
M.TURNON	Activate Program at Given Time of Day	1,X'1E'	H.REXS,66	6.2
M_TURNON	Activate Program at Given Time of Day	1,X'1E'	H.REXS,66	7.2
M.TYPE	System Console Type	1,X'3F'	H.IOCS,14	6.2
M_TYPE	System Console Type	1,X'3F'	H.IOCS,14	7.2
M.UNLOCK	Release Exclusive Resource Lock	2,X'45'	H.REMM,24	6.2
M_UNLOCK	Release Exclusive Resource Lock	2,X'45'	H.REMM,24	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M_UNPRIVMODE	Change Task to Unprivileged Mode	2,X'58'	H.REXS,79	7.2
M.UNSYNC	Release Synchronous Resource Lock	2,X'47'	H.REMM,26	6.2
M_UNSYNC	Release Synchronous Resource Lock	2,X'47'	H.REMM,26	7.2
M.UPRIV	Change Task to Unprivileged Mode	2,X'58'	H.REXS,79	6.2
M.UPSP	Upspace	1,X'10'	H.IOCS,20	6.2
M_UPSP	Upspace	1,X'10'	H.IOCS,20	7.2
M.USER	User Name Specification	1,X'74'	H.MONS,34	6.4
M.VADDR	Validate Address Range	2,X'59'	H.REXS,33	6.2
M_VADDR	Validate Address Range	2,X'59'	H.REXS,33	7.2
M.WAIT	Wait I/O	1,X'3C'	H.IOCS,25	6.2
M_WAIT	Wait I/O	1,X'3C'	H.IOCS,25	7.2
M.WEOF	Write EOF	1,X'38'	H.IOCS,5	6.2
M.WRIT	Write Record	1,X'32'	H.IOCS,4	6.2
M_WRITE	Write Record	1,X'32'	H.IOCS,4	7.2
M_WRITEEOF	Write EOF	1,X'38'	H.IOCS,5	7.2
M.XBRKR	Exit from Task Interrupt Level	1,X'70'	H.REXS,48	6.2
M_XBRKR	Exit from Task Interrupt Level	N/A	N/A	7.2
M.XIEA	No-wait I/O End-action Return	1,X'2C'	H.IOCS,34	6.2
M_XIEA	No-wait I/O End-action Return	N/A	N/A	7.2
M.XMEA	Exit from Message End-action Routine	1,X'7E'	H.REXS,50	6.2
M_XMEA	Exit from Message End-action Routine	N/A	N/A	7.2

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
M.XMSGR	Exit from Message Receiver	1,X'5E'	H.REXS,39	6.2
M_XMSGR	Exit from Message Receiver	N/A	N/A	7.2
M.XREA	Exit from Run Request End-action Routine	1,X'7F'	H.REXS,51	6.2
M_XREA	Exit from Run Request End-action Routine	N/A	N/A	7.2
M.XRUNR	Exit Run Receiver	1,X'7D'	H.REXS,49	6.2
M_XRUNR	Exit Run Receiver	N/A	N/A	7.2
M.XTIME	Task CPU Execution Time	1,X'2D'	H.REXS,65	6.2
M_XTIME	Task CPU Execution Time	1,X'2D'	H.REXS,65	7.2
N/A	Allocate File Space	N/A	H.VOMM,19	6.3
N/A	Allocate Resource Descriptor	N/A	H.VOMM,17	6.3
N/A	Create Temporary File	N/A	H.VOMM,24	6.3
N/A	Deallocate File Space	N/A	H.VOMM,20	6.3
N/A	Deallocate Resource Descriptor	N/A	H.VOMM,18	6.3
N/A	Debug Link Service	1,X'66'	H.REXS,42	6.3
N/A	Debug Link Service-Base Mode	1,X'66'	H.REXS,42	7.3
N/A	Eject/Purge Routine	1,X'0D'	H.IOCS,22	6.3
N/A	Eject/Purge Routine-Base Mode	1,X'0D'	H.IOCS,22	7.3
N/A	Erase or Punch Trailer	1,X'3E'	H.IOCS,21	6.3
N/A	Erase or Punch Trailer - Base Mode	1,X'3E'	H.IOCS,21	7.3
N/A	Execute Channel Program	1,X'25'	H.IOCS,10	6.3
N/A	Execute Channel Program - Base Mode	1,X'25'	H.IOCS,10	7.3

Macro	Description	SVC	Module, E.P.	Volume I Ref.Manual Section
N/A	Get Extended Memory Array	2,X'7F'	H.MEMM,14	6.3
N/A	Get Extended Memory Array - Base Mode	2,X'7F'	H.MEMM,14	7.3
N/A	Read/Write Authorization File	N/A	H.VOMM,25	6.3
N/A	Release FHD Port	1,X'27'	H.IOCS,27	6.3
N/A	Release FHD Port - Base Mode	1,X'27'	H.IOCS,27	7.3
N/A	Reserve FHD Port	1,X'26'	H.IOCS,24	6.3
N/A	Reserve FHD Port- Base Mode	1,X'26'	H.IOCS,24	7.3
N/A	Reserved for Interactive Debugger	2,X'56'	H.REXS,30	N/A
N/A	Reserved for Rapid File Allocation:			N/A
	Zero MDT	2,X'AA'	H.MDT,1	
	Locate/Read MDT Entry	2,X'AB'	H.MDT,2	
	Update/Create MDT Entry	2,X'AC'	H.MDT,3	
	Delete MDT Entry	2,X'AD'	H.MDT,4	
N/A	Set Tabs in UDT	1,X'59'	H.TSM,5	N/A
N/A	TSM Task Detach	1,X'20'	H.TSM,3	N/A

B.2 Alphabetic Listing

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Abort Self	M.BORT	1,X'57'	H.REXS,20	6.2
	M_BORT	1,X'57'	H.REXS,20	7.2
Abort Specified Task	M.BORT	1,X'56'	H.REXS,19	6.2
	M_BORT	1,X'56'	H.REXS,19	7.2
Abort With Extended Message	M.BORT	1,X'62'	H.REXS,28	6.2
	M_BORT	1,X'62'	H.REXS,28	7.2
Acquire Current Date/Time in ASCII Format	M.QATIM	2,X'50'	H.REXS,74	6.2
	M_QATIM	2,X'50'	H.REXS,74	7.2
Acquire Current Date/Time in Binary Format	M.BTIM	2,X'50'	H.REXS,74	6.2
	M_BTIM	2,X'50'	H.REXS,74	7.2
Acquire Current Date/Time in Byte Binary Format	M.BBTIM	2,X'50'	H.REXS,74	6.2
	M_BBTIM	2,X'50'	H.REXS,74	7.2
Acquire System Date/Time in Any Format	M.GTIM	2,X'50'	H.REXS,74	6.2
	M_GTIM	2,X'50'	H.REXS,74	7.2
Activate Program at Given Time of Day	M.TURNON	1,X'1E'	H.REXS,66	6.2
	M_TURNON	1,X'1E'	H.REXS,66	7.2
Activate Task	M.ACTV	1,X'52'	H.REXS,15	6.2
	M_ACTV	1,X'52'	H.REXS,15	7.2
Activate Task Interrupt	M.INT	1,X'6F'	H.REXS,47	6.2
	M_INT	1,X'6F'	H.REXS,47	7.2
Advance File	M.FWRD	1,X'34'	H.IOCS,8	6.2
	M_ADVANCE	1,X'34'	H.IOCS,8	7.2
Advance Record	M.FWRD	1,X'33'	H.IOCS,7	6.2
	M_ADVANCE	1,X'33'	H.IOCS,7	7.2
Allocate File or Peripheral Device	M.ALOC	1,X'40'	H.MONS,21	6.4
Allocate File Space	N/A	N/A	H.VOMM,19	6.3
Allocate Resource Descriptor	N/A	N/A	H.VOMM,17	6.3
Arithmetic Exception Inquiry	M.TSTE	1,X'4D'	H.REXS,23	6.2
	M_TSTE	1,X'4D'	H.REXS,23	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Assign and Allocate Resource	M.ASSN M_ASSIGN	2,X'52' 2,X'52'	H.REXS,21 H.REXS,21	6.2 7.2
Backspace File	M.BACK M_BACKSPACE	1,X'36' 1,X'36'	H.IOCS,19 H.IOCS,19	6.2 7.2
Backspace Record	M.BACK M_BACKSPACE	1,X'35' 1,X'35'	H.IOCS,9 H.IOCS,9	6.2 7.2
Batch Job Entry	M.BATCH M_BATCH	2,X'55' 2,X'55'	H.REXS,27 H.REXS,27	6.2 7.2
Break/Task Interrupt Link/Unlink	M.BRK M_BRK	1,X'6E' 1,X'6E'	H.REXS,46 H.REXS,46	6.2 7.2
Change Defaults	M.DEFT M_DEFT	2,X'27' 2,X'27'	H.VOMM,8 H.VOMM,8	6.2 7.2
Change Priority Level	M.PRIL M_PRIL	1,X'4A' 1,X'4A'	H.REXS,9 H.REXS,9	6.2 7.2
Change Task to Unprivileged Mode	M.UPRIV M_UNPRIVMODE	2,X'58' 2,X'58'	H.REXS,79 H.REXS,79	6.2 7.2
Change Temporary File to Permanent	M.PERM	1,X'76'	H.FISE,13	6.4
Change Temporary File to Permanent File	M.TEMPER M_TEMPFILETOPERM	2,X'28' 2,X'28'	H.VOMM,9 H.VOMM,9	6.2 7.2
Close File	M.CLSE M_CLSE	1,X'39' 1,X'39'	H.IOCS,23 H.IOCS,23	6.2 7.2
Close Resource	M.CLOSER M_CLOSER	2,X'43' 2,X'43'	H.REMM,22 H.REMM,22	6.2 7.2
Connect Task to Interrupt	M.CONN M_CONN	1,X'4B' 1,X'4B'	H.REXS,10 H.REXS,10	6.2 7.2
Convert ASCII Date/Time to Byte Binary Format	M.CONABB M_CONABB	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert ASCII Date/Time to Standard Binary	M.CONASB M_CONASB	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert ASCII Decimal to Binary	M.CONADB M_CONADB	1,X'28' 1,X'28'	H.TSM,7 H.TSM,7	6.2 7.2
Convert ASCII Hex to Binary	M.CONAHB M_CONAHB	1,X'29' 1,X'29'	H.TSM,8 H.TSM,8	6.2 7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Convert Binary Date/Time to ASCII Format	M.CONBAF M_CONBAF	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Binary Date/Time to Byte Binary	M.CONBBY M_CONBBY	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Binary to ASCII Decimal	M.CONBAD M_CONBAD	1,X'2A' 1,X'2A'	H.TSM,9 H.TSM,9	6.2 7.2
Convert Binary to ASCII Hex	M.CONBAH M_CONBAH	1,X'2B' 1,X'2B'	H.TSM,10 H.TSM,10	6.2 7.2
Convert Byte Binary Date/Time to ASCII	M.CONBBA M_CONBBA	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Byte Binary Date/Time to Binary	M.CONBYB M_CONBYB	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Pathname to Pathname Block	M.PNAMB M_PNAMB	2,X'2E' 2,X'2E'	H.VOMM,15 H.VOMM,15	6.2 7.2
Convert System Date/Time Format	M.CTIM M_CTIM	2,X'51' 2,X'51'	H.REXS,75 H.REXS,75	6.2 7.2
Convert Time	M_CONVERTTIME	2,X'51'	H.REXS,75	7.2
Create Directory	M.DIR M_DIR	2,X'23' 2,X'23'	H.VOMM,4 H.VOMM,4	6.2 7.2
Create File Control Block	M.DFCB	N/A	N/A	5.9.1
Create File Control Block	M_CREATEFCB	N/A	N/A	7.2
Create Memory Partition	M.MEM M_MEM	2,X'22' 2,X'22'	H.VOMM,3 H.VOMM,3	6.2 7.2
Create Permanent File	M.CREATE	1,X'75'	H.FISE,12	6.4
Create Permanent File	M.CPERM M_CREATEP	2,X'20' 2,X'20'	H.VOMM,1 H.VOMM,1	6.2 7.2
Create Temporary File	M.TEMP M_CREATET	2,X'21' 2,X'21'	H.VOMM,2 H.VOMM,2	6.2 7.2
Create Temporary File	N/A	N/A	H.VOMM,24	6.3

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Create Timer Entry	M.SETT	1,X'45'	H.REXS,4	6.2
	M_SETT	1,X'45'	H.REXS,4	7.2
Date and Time Inquiry	M.DATE	1,X'15'	H.REXS,70	6.2
	M_DATE	1,X'15'	H.REXS,70	7.2
Deallocate File or Peripheral Device	M.DALC	1,X'41'	H.MONS,22	6.4
Deallocate File Space	N/A	N/A	H.VOMM,20	6.3
Deallocate Resource Descriptor	N/A	N/A	H.VOMM,18	6.3
Deassign and Deallocate Resource	M.DASN	2,X'53'	H.REXS,22	6.2
	M_DEASSIGN	2,X'53'	H.REXS,22	7.2
Debug Link Service	N/A	1,X'66'	H.REXS,42	6.3
Debug Link Service-Base Mode	N/A	1,X'66'	H.REXS,42	7.3
Delete Permanent File or Non-SYSGEN Memory Partition	M.DELETE	1,X'77'	H.FISE,14	6.4
Delete Resource	M.DELR	2,X'24'	H.VOMM,5	6.2
	M_DELETER	2,X'24'	H.VOMM,5	7.2
Delete Task	M.DELTSK	1,X'5A'	H.REXS,31	6.2
	M_DELTSK	1,X'5A'	H.REXS,31	7.2
Delete Timer Entry	M.DLTT	1,X'47'	H.REXS,6	6.2
	M_DLTT	1,X'47'	H.REXS,6	7.2
Disable Message Task Interrupt	M.DSMI	1,X'2E'	H.REXS,57	6.2
	M_DSMI	1,X'2E'	H.REXS,57	7.2
Disable User Break Interrupt	M.DSUB	1,X'12'	H.REXS,73	6.2
	M_DSUB	1,X'12'	H.REXS,73	7.2
Disconnect Task from Interrupt	M.DISCON	1,X'5D'	H.REXS,38	6.2
	M_DISCON	1,X'5D'	H.REXS,38	7.2
Dismount Volume	M.DMOUNT	2,X'4A'	H.REMM,19	6.2
	M_DISMOUNT	2,X'4A'	H.REMM,19	7.2
Eject/Purge Routine	N/A	1,X'0D'	H.IOCS,22	6.3
Eject/Purge Routine-Base Mode	N/A	1,X'0D'	H.IOCS,22	7.3
Enable Message Task Interrupt	M.ENMI	1,X'2F'	H.REXS,58	6.2
	M_ENMI	1,X'2F'	H.REXS,58	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Enable User Break Interrupt	M.ENUB	1,X'13'	H.REXS,72	6.2
	M_ENUB	1,X'13'	H.REXS,72	7.2
End Action Wait	M.EAWAIT	1,X'1D'	H.EXEC,40	6.2
	M_AWAITACTION	1,X'1D'	H.EXEC,40	7.2
Erase or Punch Trailer	N/A	1,X'3E'	H.IOCS,21	6.3
Erase or Punch Trailer - Base Mode	N/A	1,X'3E'	H.IOCS,21	7.3
Exclude Memory Partition	M.EXCLUDE	2,X'41'	H.REMM,14	6.2
Exclude Shared Image	M_EXCLUDE	2,X'41'	H.REMM,14	7.2
Execute Channel Program	N/A	1,X'25'	H.IOCS,10	6.3
Execute Channel Program - Base Mode	N/A	1,X'25'	H.IOCS,10	7.3
Execute Channel Program File Control Block	M_CHANPROGFCB	N/A	N/A	7.2
Exit from Message End-action Routine	M.XMEA	1,X'7E'	H.REXS,50	6.2
	M_XMEA	N/A	N/A	7.2
Exit from Message Receiver	M.XMSGR	1,X'5E'	H.REXS,39	6.2
	M_XMSGR	N/A	N/A	7.2
Exit from Run Request End-action Routine	M.XREA	1,X'7F'	H.REXS,51	6.2
	M_XREA	N/A	N/A	7.2
Exit from Task Interrupt Level	M.BRKXIT	1,X'70'	H.REXS,48	6.2
	M _BRKXIT	N/A	N/A	7.2
	M.XBRKR	1,X'70'	H.REXS,48	6.2
	M _XBRKR	N/A	N/A	7.2
Exit Run Receiver	M.XRUNR	1,X'7D'	H.REXS,49	6.2
	M_XRUNR	N/A	N/A	7.2
Exit With Status	M_EXTSTS	2,X'5F'	H.REXS,86	7.2
Extend File	M.EXTD	2,X'25'	H.VOMM,6	6.2
	M_EXTENDFILE	2,X'25'	H.VOMM,6	7.2
Free Dynamic Extended Indexed Data Space	M.FD	1,X'6A'	H.REMM,9	6.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Free Dynamic Task Execution Space	M.FE	1,X'68'	H.REMM,11	6.2
Free Memory in Byte Increments	M.MEMFRE	2,X'4C'	H.REMM,29	6.2
	M_FREEMEMBYTES	2,X'4C'	H.REMM,29	7.2
Free Shared Memory	M.EXCL	1,X'79'	H.ALOC,14	6.4
Get Address Limits	M.GADRL	1,X'65'	H.REXS,41	6.2
Get Address Limits	M.GADRL2	2,X'7B'	H.REXS,80	6.2
Get Base Mode Task Address Limits	M_LIMITS	2,X'5D'	H.REXS,84	7.2
Get Command Line	M.CMD	2,X'61'	H.REXS,88	6.2
	M_CMD	2,X'61'	H.REXS,88	7.2
Get Current Date and Time	M_GETTIME	2,X'50'	H.REXS,74	7.2
Get Device Mnemonic or Type Code	M.DEVID	1,X'14'	H.REXS,71	6.2
	M_DEVID	1,X'14'	H.REXS,71	7.2
Get Dynamic Extended Data Space	M.GD	1,X'69'	H.REMM,8	6.2
Get Dynamic Extended Discontiguous Data Space	M.GDD	2,X'7C'	H.MEMM,9	6.2
Get Dynamic Task Execution Space	M.GE	1,X'67'	H.REMM,10	6.2
Get Extended Memory Array	N/A	2,X'7F'	H.MEMM,14	6.3
Get Extended Memory Array - Base Mode	N/A	2,X'7F'	H.MEMM,14	7.3
Get Memory in Byte Increments	M.MEMB	2,X'4B'	H.REMM,28	6.2
	M_GETMEMBYTES	2,X'4B'	H.REMM,28	7.2
Get Message Parameters	M.GMSGP	1,X'7A'	H.REXS,35	6.2
	M_GMSGP	1,X'7A'	H.REXS,35	7.2
Get Real Physical Address	M.RADDR	1,X'0E'	H.REXS,90	6.2
	M_RADDR	1,X'0E'	H.REXS,90	7.2
Get Run Parameters	M.GRUNP	1,X'7B'	H.REXS,36	6.2
	M_GRUNP	1,X'7B'	H.REXS,36	7.2
Get Shared Memory	M.INCL	1,X'72'	H.ALOC,13	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Get Task Environment	M.ENVRMT	2,X'5E'	H.REXS,85	6.2
	M_ENVRMT	2,X'5E'	H.REXS,85	7.2
Get Task Number	M.ID	1,X'64'	H.REXS,32	6.2
	M_ID	1,X'64'	H.REXS,32	7.2
	M.MYID	1,X'64'	H.REXS,32	6.2
	M_MYID	1,X'64'	H.REXS,32	7.2
Get Terminal Function Definition	M.GETDEF	2,X'7A'	H.TSM,15	6.2
	M_GETDEF	2,X'7A'	H.TSM,15	7.2
Get TSA Start Address	M.GTSAD	2,X'7D'	H.REXS,91	6.2
	M_GTSAD	2,X'7D'	H.REXS,91	7.2
Get User Context	M_GETCTX	2,X'70'	H.EXEC,41	7.2
Include Memory Partition	M.INCLUDE	2,X'40'	H.REMM,12	6.2
Include Shared Image	M_INCLUDE	2,X'40'	H.REMM,12	7.2
Load and Execute Interactive Debugger	M.DEBUG	1,X'63'	H.REXS,29	6.2
	M_DEBUG	1,X'63'	H.REXS,29	7.2
Load Overlay Segment	M.OLAY	1,X'50'	H.REXS,13	6.2
Load and Execute Overlay		1,X'51'	H.REXS,14	6.2
Log Resource or Directory	M.LOGR	2,X'29'	H.VOMM,10	6.2
	M_LOGR	2,X'29'	H.VOMM,10	7.2
Memory Address Inquiry	M.ADRS	1,X'44'	H.REXS,3	6.2
	M_ADRS	1,X'44'	H.REXS,3	7.2
Memory Dump Request	M.DUMP	1,X'4F'	H.REXS,12	6.2
	M_DUMP	1,X'4F'	H.REXS,12	7.2
Modify Descriptor	M.MOD	2,X'2A'	H.VOMM,11	6.2
	M_MOD	2,X'2A'	H.VOMM,11	7.2
Modify Descriptor User Area	M.MODU	2,X'31'	H.VOMM,26	6.2
	M_MODU	2,X'31'	H.VOMM,26	7.2
Mount Volume	M.MOUNT	2,X'49'	H.REMM,17	6.2
	M_MOUNT	2,X'49'	H.REMM,17	7.2
Move Data to User Address	M.MOVE	2,X'62'	H.REXS,89	6.2
	M_MOVE	2,X'62'	H.REXS,89	7.2
No-wait I/O End-action Return	M.XIEA	1,X'2C'	H.IOCS,34	6.2
	M_XIEA	N/A	N/A	7.2
Open File	M.FILE	1,X'30'	H.IOCS,1	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Open Resource	M.OPENR	2,X'42'	H.REMM,21	6.2
	M_OPENR	2,X'42'	H.REMM,21	7.2
Parameter Task Activation	M.PTSK	1,X'5F'	H.REXS,40	6.2
	M_PTSK	1,X'5F'	H.REXS,40	7.2
Permanent File Address Inquiry	M.FADD	1,X'43'	H.MONS,2	6.4
Permanent File Log	M.LOG	1,X'73'	H.MONS,33	6.4
Physical Device Inquiry	M.PDEV	1,X'42'	H.MONS,1	6.4
Physical Memory Read	M.OSREAD	2,X'7E'	H.REXS,93	6.2
	M_OSREAD	2,X'7E'	H.REXS,93	7.2
Physical Memory Write	M.OSWRIT	2,X'AF'	H.REXS,94	6.2
	M_OSWRIT	2,X'AF'	H.REXS,94	7.2
Program Hold Request	M.HOLD	1,X'58'	H.REXS,25	6.2
	M_HOLD	1,X'58'	H.REXS,25	7.2
Put User Context	M_PUTCTX	2,X'71'	H.EXEC,42	7.2
Read Descriptor	M.LOC	2,X'2C'	H.VOMM,13	6.2
	M_READD	2,X'2C'	H.VOMM,13	7.2
Read Record	M.READ	1,X'31'	H.IOCS,3	6.2
	M_READ	1,X'31'	H.IOCS,3	7.2
Read/Write Authorization File	N/A	N/A	H.VOMM,25	6.3
Receive Message Link Address	M.RCVR	1,X'6B'	H.REXS,43	6.2
	M_RCVR	1,X'6B'	H.REXS,43	7.2
Reconstruct Pathname	M.PNAM	2,X'2F'	H.VOMM,16	6.2
	M_CONSTRUCTPATH	2,X'2F'	H.VOMM,16	7.2
Reformat RRS Entry	M.NEWRRS	2,X'54'	H.REXS,76	6.2
Reinstate Privilege Mode to Privilege Task	M.PRIV	2,X'57'	H.REXS,78	6.2
	M_PRIVMODE	2,X'57'	H.REXS,78	7.2
Release Channel Reservation	M.RRES	1,X'3B'	H.IOCS,13	6.2
	M_RRES	1,X'3B'	H.IOCS,13	7.2
Release Dual-ported Disc/Set Dual-channel ACM Mode	M.RELP	1,X'27'	H.IOCS,27	6.2
	M_RELP	1,X'27'	H.IOCS,27	7.2
Release Exclusive File Lock	M.FXLR	1,X'22'	H.FISE,23	6.4
Release Exclusive Resource Lock	M.UNLOCK	2,X'45'	H.REMM,24	6.2
	M_UNLOCK	2,X'45'	H.REMM,24	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Release FHD Port	N/A	1,X'27'	H.IOCS,27	6.3
Release FHD Port-Base Mode	N/A	1,X'27'	H.IOCS,27	7.3
Release Synchronization File Lock	M.FSLR	1,X'24'	H.FISE,25	6.4
Release Synchronous Resource Lock	M.UNSYNC	2,X'47'	H.REMM,26	6.2
	M_UNSYNC	2,X'47'	H.REMM,26	7.2
Rename File	M.RENAM	2,X'2D'	H.VOMM,14	6.2
	M_RENAME	2,X'2D'	H.VOMM,14	7.2
Replace Permanent File	M.REPLAC	2,X'30'	H.VOMM,23	6.2
	M_REPLACE	2,X'30'	H.VOMM,23	7.2
Reserve Channel	M.RSRV	1,X'3A'	H.IOCS,12	6.2
	M_RSRV	1,X'3A'	H.IOCS,12	7.2
Reserve Dual-ported Disc/Set Single-channel ACM Mode	M.RESP	1,X'26'	H.IOCS,24	6.2
	M_RESP	1,X'26'	H.IOCS,24	7.2
Reserve FHD Port	N/A	1,X'26'	H.IOCS,24	6.3
Reserve FHD Port-Base Mode	N/A	1,X'26'	H.IOCS,24	7.3
Reserved for Interactive Debugger	N/A	2,X'56'	H.REXS,30	N/A
Reserved for Rapid File Allocation:	N/A			N/A
Zero MDT		2,X'AA'	H.MDT,1	
Locate/Read MDT Entry		2,X'AB'	H.MDT,2	
Update/Create MDT Entry		2,X'AC'	H.MDT,3	
Delete MDT Entry		2,X'AD'	H.MDT,4	
Reset Option Lower	M.ROPL	2,X'78'	H.TSM,14	6.2
	M_ROPL	2,X'78'	H.TSM,14	7.2
Resource Inquiry	M.INQUIRY	2,X'48'	H.REMM,27	6.2
	M_INQUIRER	2,X'48'	H.REMM,27	7.2
Resourcemark Lock	M.RSML	1,X'19'	H.REXS,62	6.2
	M_RSML	1,X'19'	H.REXS,62	7.2
Resourcemark Unlock	M.RSMU	1,X'1A'	H.REXS,63	6.2
	M_RSMU	1,X'1A'	H.REXS,63	7.2

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Resume Task Execution	M.SUME M_SUME	1,X'53' 1,X'53'	H.REXS,16 H.REXS,16	6.2 7.2
Rewind File	M.RWND M_REWIND	1,X'37' 1,X'37'	H.IOCS,2 H.IOCS,2	6.2 7.2
Rewrite Descriptor	M.REWRIT M_REWRIT	2,X'2B' 2,X'2B'	H.VOMM,12 H.VOMM,12	6.2 7.2
Rewrite Descriptor User Area	M.REWRTU M_REWRTU	2,X'32' 2,X'32'	H.VOMM,27 H.VOMM,27	6.2 7.2
Scan Terminal Input Buffer	M.TSCAN M_TSCAN	1,X'5B' 1,X'5B'	H.TSM,2 H.TSM,2	6.2 7.2
Send Message to Specified Task	M.SMSGR M_SMSGR	1,X'6C' 1,X'6C'	H.REXS,44 H.REXS,44	6.2 7.2
Send Run Request to Specified Task	M.SRUNR M_SRUNR	1,X'6D' 1,X'6D'	H.REXS,45 H.REXS,45	6.2 7.2
Set Asynchronous Task Interrupt	M.ASYNCH M_ASYNCH	1,X'1C' 1,X'1C'	H.REXS,68 H.REXS,68	6.2 7.2
Set Exception Handler	M_SETEXA	2,X'5C'	H.REXS,83	7.2
Set Exception Return Address	M_SETERA	2,X'79'	H.REXS,81	7.2
Set Exclusive File Lock	M.FXLS	1,X'21'	H.FISE,22	6.4
Set Exclusive Resource Lock	M.LOCK M_LOCK	2,X'44' 2,X'44'	H.REMM,23 H.REMM,23	6.2 7.2
Set IPU Bias	M.IPUBS M_IPUBS	2,X'5B' 2,X'5B'	H.REXS,82 H.REXS,82	6.2 7.2
Set Option Lower	M.SOPL M_SOPL	2,X'77' 2,X'77'	H.TSM,13 H.TSM,13	6.2 7.2
Set Synchronization File Lock	M.FSLS	1,X'23'	H.FISE,24	6.4
Set Synchronous Resource Lock	M.SETSYNC M_SETSYNC	2,X'46' 2,X'46'	H.REMM,25 H.REMM,25	6.2 7.2
Set Synchronous Task Interrupt	M.SYNCH M_SYNCH	1,X'1B' 1,X'1B'	H.REXS,67 H.REXS,67	6.2 7.2
Set Tabs in UDT	N/A	1,X'59'	H.TSM,5	N/A

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Set User Abort Receiver Address	M.SUAR M_SUAR	1,X'60' 1,X'60'	H.REXS,26 H.REXS,26	6.2 7.2
Set User Status Word	M.SETS M_SETS	1,X'48' 1,X'48'	H.REXS,7 H.REXS,7	6.2 7.2
Share Memory with Another Task	M.SHARE	1,X'71'	H.ALOC,12	6.4
Submit Job from Disc File	M.CDJS	1,X'61'	H.MONS,27	6.4
Suspend/Resume	M.SURE M_SURE	5,X'00' 5,X'00'	N/A N/A	6.2 7.2
Suspend Task Execution	M.SUSP M_SUSP	1,X'54' 1,X'54'	H.REXS,17 H.REXS,17	6.2 7.2
System Console Type	M.TYPE M_TYPE	1,X'3F' 1,X'3F'	H.IOCS,14 H.IOCS,14	6.2 7.2
System Console Wait	M.CWAT M_CWAT	1,X'3D' 1,X'3D'	H.IOCS,26 H.IOCS,26	6.2 7.2
Task CPU Execution Time	M.XTIME M_XTIME	1,X'2D' 1,X'2D'	H.REXS,65 H.REXS,65	6.2 7.2
Task Option Doubleword Inquiry	M.PGOD M_OPTIONDWORD	2,X'C0' 2,X'C0'	H.REXS,95 H.REXS,95	6.2 7.2
Task Option Word Inquiry	M.PGOW M_OPTIONWORD	1,X'4C' 1,X'4C'	H.REXS,24 H.REXS,24	6.2 7.2
Terminate Task Execution	M.EXIT M_EXIT	1,X'55' 1,X'55'	H.REXS,18 H.REXS,18	6.2 7.2
Test Timer Entry	M.TSTT M_TSTT	1,X'46' 1,X'46'	H.REXS,5 H.REXS,5	6.2 7.2
Test User Status Word	M.TSTS M_TSTS	1,X'49' 1,X'49'	H.REXS,8 H.REXS,8	6.2 7.2
Time-of-Day Inquiry	M.TDAY M_TDAY	1,X'4E' 1,X'4E'	H.REXS,11 H.REXS,11	6.2 7.2
Trap On-line User's Task	M.TBRKON M_TBRKON	1,X'5C' 1,X'5C'	H.TSM,6 H.TSM,6	6.2 7.2
Truncate File	M.TRNC M_TRUNCATE	2,X'26' 2,X'26'	H.VOMM,7 H.VOMM,7	6.2 7.2
TSM Procedure Call	M.TSMPC M_TSMPC	2,X'AE' 2,X'AE'	H.TSM,17 H.TSM,17	6.2 7.2
TSM Task Detach	N/A	1,X'20'	H.TSM,3	N/A
Unlock and Dequeue Shared Memory	M.SMULK	1,X'1F'	H.ALOC,19	6.4

Description	Macro	SVC	Module, E.P.	Volume I Ref.Manual Section
Upspace	M.UPSP	1,X'10'	H.IOCS,20	6.2
	M_UPSP	1,X'10'	H.IOCS,20	7.2
User Name Specification	M.USER	1,X'74'	H.MONS,34	6.4
Validate Address Range	M.VADDR	2,X'59'	H.REXS,33	6.2
	M_VADDR	2,X'59'	H.REXS,33	7.2
Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	M.ANYW	1,X'7C'	H.REXS,37	6.2
	M_ANYWAIT	1,X'7C'	H.REXS,37	7.2
Wait I/O	M.WAIT	1,X'3C'	H.IOCS,25	6.2
	M_WAIT	1,X'3C'	H.IOCS,25	7.2
Write EOF	M.WEOF	1,X'38'	H.IOCS,5	6.2
	M_WRITEEOF	1,X'38'	H.IOCS,5	7.2
Write Record	M.WRIT	1,X'32'	H.IOCS,4	6.2
	M_WRITE	1,X'32'	H.IOCS,4	7.2

B.3 SVC Listing

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00-0A	Reserved			
0B	Reserved for Vector Processor			
0C	Reserved			
0D	Eject/Purge Routine	H.IOCS,22	N/A	6.3
	Eject/Purge Routine - Base Mode	H.IOCS,22	N/A	7.3
0E	Get Real Physical Address	H.REXS,90	M.RADDR M_RADDR	6.2 7.2
0F	Reserved for Vector Processor		N/A	N/A
10	Upspace	H.IOCS,20	M.UPSP M_UPSP	6.2 7.2
11	Reserved			
12	Disable User Break Interrupt	H.REXS,73	M.DSUB M_DSUB	6.2 7.2
13	Enable User Break Interrupt	H.REXS,72	M.ENUB M_ENUB	6.2 7.2
14	Get Device Mnemonic or Type Code	H.REXS,71	M.DEVID M_DEVID	6.2 7.2
15	Date and Time Inquiry	H.REXS,70	M.DATE M_DATE	6.2 7.2
16	ADI Maximum IOCBs	N/A	M.ADIMAX	N/A
17	ADI I/O	N/A	M.ADIO	N/A
18	ADI EAI	N/A	M.ADIEAI	N/A
19	Resourcemark Lock	H.REXS,62	M.RSML M_RSML	6.2 7.2
1A	Resourcemark Unlock	H.REXS,63	M.RSMU M_RSMU	6.2 7.2
1B	Set Synchronous Task Interrupt	H.REXS,67	M.SYNCH M_SYNCH	6.2 7.2
1C	Set Asynchronous Task Interrupt	H.REXS,68	M.ASYNCH M_ASYNCH	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
1D	End Action Wait	H.EXEC,40	M.EAWAIT M_AWAITACTION	6.2 7.2
1E	Activate Program at Given Time of Day	H.REXS,66	M.TURNON M_TURNON	6.2 7.2
1F	Unlock and Dequeue Shared Memory	H.ALOC,19	M.SMULK	6.4
20	TSM Task Detach	H.TSM,3	N/A	N/A
21	Set Exclusive File Lock	H.FISE,22	M.FXLS	6.4
22	Release Exclusive File Lock	H.FISE,23	M.FXLR	6.4
23	Set Synchronization File Lock	H.FISE,24	M.FSLS	6.4
24	Release Synchronization File Lock	H.FISE,25	M.FSLR	6.4
25	Execute Channel Program	H.IOCS,10	N/A	6.3
	Execute Channel Program - Base Mode	H.IOCS,10	N/A	7.3
26	ReserveFHD Port Reserve FHD Port - Base Mode Reserve Dual-ported Disc/Set Single-channel ACM Mode	H.IOCS,24	N/A N/A M.RESP M_RESP	6.3 7.3 6.2 7.2
27	Release FHD Port Release FHD Port - Base Mode Release Dual-ported Disc/Set Dual-channel ACM Mode	H.IOCS,27	N/A N/A M.RELP M_RELP	6.3 7.3 6.2 7.2
28	Convert ASCII Decimal to Binary	H.TSM,7	M.CONADB M_CONADB	6.2 7.2
29	Convert ASCII Hex to Binary	H.TSM,8	M.CONAHB M_CONAHB	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
2A	Convert Binary to ASCII Decimal	H.TSM,9	M.CONBAD M_CONBAD	6.2 7.2
2B	Convert Binary to ASCII Hex	H.TSM,10	M.CONBAH M_CONBAH	6.2 7.2
2C	No-wait I/O End-action Return	H.IOCS,34	M.XIEA	6.2
2D	Task CPU Execution Time	H.REXS,65	M.XTIME M_XTIME	6.2 7.2
2E	Disable Message Task Interrupt	H.REXS,57	M.DSMI M_DSMI	6.2 7.2
2F	Enable Message Task Interrupt	H.REXS,58	M.ENMI M_ENMI	6.2 7.2
30	Open File	H.IOCS,1	M.FILE	6.4
31	Read Record	H.IOCS,3	M.READ M_READ	6.2 7.2
32	Write Record	H.IOCS,4	M.WRIT M_WRITE	6.2 7.2
33	Advance Record	H.IOCS,7	M.FWRD M_ADVANCE	6.2 7.2
34	Advance File	H.IOCS,8	M.FWRD M_ADVANCE	6.2 7.2
35	Backspace Record	H.IOCS,9	M.BACK M_BACKSPACE	6.2 7.2
36	Backspace File	H.IOCS,19	M.BACK M_BACKSPACE	6.2 7.2
37	Rewind File	H.IOCS,2	M.RWND M_REWIND	6.2 7.2
38	Write EOF	H.IOCS,5	M.WEOF M_WRITEEOF	6.2 7.2
39	Close File	H.IOCS,23	M.CLSE M_CLSE	6.2 7.2
3A	Reserve Channel	H.IOCS,12	M.RSRV M_RSRV	6.2 7.2
3B	Release Channel Reservation	H.IOCS,13	M.RRES M_RRES	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
3C	Wait I/O	H.IOCS,25	M.WAIT M_WAIT	6.2 7.2
3D	System Console Wait	H.IOCS,26	M.CWAT M_CWAT	6.2 7.2
3E	Erase or Punch Trailer	H.IOCS,21	N/A	6.3
	Erase or Punch Trailer - Base Mode	H.IOCS,21	N/A	7.3
3F	System Console Type	H.IOCS,14	M.TYPE M_TYPE	6.2 7.2
40	Allocate File or Peripheral Device	H.MONS,21	M.ALOC	6.4
41	Deallocate File or Peripheral Device	H.MONS,22	M.DALC	6.4
42	Physical Device Inquiry	H.MONS,1	M.PDEV	6.4
43	Permanent File Address Inquiry	H.MONS,2	M.FADD	6.4
44	Memory Address Inquiry	H.REXS,3	M.ADRS M_ADRS	6.2 7.2
45	Create Timer Entry	H.REXS,4	M.SETT M_SETT	6.2 7.2
46	Test Timer Entry	H.REXS,5	M.TSTT M_TSTT	6.2 7.2
47	Delete Timer Entry	H.REXS,6	M.DLTT M_DLTT	6.2 7.2
48	Set User Status Word	H.REXS,7	M.SETS M_SETS	6.2 7.2
49	Test User Status Word	H.REXS,8	M.TSTS M_TSTS	6.2 7.2
4A	Change Priority Level	H.REXS,9	M.PRIL M_PRIL	6.2 7.2
4B	Connect Task to Interrupt	H.REXS,10	M.CONN M_CONN	6.2 7.2
4C	Task Option Word Inquiry	H.REXS,24	M.PGOW M_OPTIONWORD	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
4D	Arithmetic Exception Inquiry	H.REXS,23	M.TSTE M_TSTE	6.2 7.2
4E	Time-of-Day Inquiry	H.REXS,11	M.TDAY M_TDAY	6.2 7.2
4F	Memory Dump Request	H.REXS,12	M.DUMP M_DUMP	6.2 7.2
50	Load Overlay Segment	H.REXS,13	M.OLAY	6.2
51	Load and Execute Overlay	H.REXS,14	M.OLAY	6.2
52	Activate Task	H.REXS,15	M.ACTV M_ACTV	6.2 7.2
53	Resume Task Execution	H.REXS,16	M.SUME M_SUME	6.2 7.2
54	Suspend Task Execution	H.REXS,17	M.SUSP M_SUSP	6.2 7.2
55	Terminate Task Execution	H.REXS,18	M.EXIT M_EXIT	6.2 7.2
56	Abort Specified Task	H.REXS,19	M.BORT M_BORT	6.2 7.2
57	Abort Self	H.REXS,20	M.BORT M_BORT	6.2 7.2
58	Program Hold Request	H.REXS,25	M.HOLD M_HOLD	6.2 7.2
59	Set Tabs in UDT	H.TSM,5	N/A	N/A
5A	Delete Task	H.REXS,31	M.DELTSK M_DELTSK	6.2 7.2
5B	Scan Terminal Input Buffer	H.TSM,2	M.TSCAN M_TSCAN	6.2 7.2
5C	Trap On-line User's Task	H.TSM,6	M.TBRKON M_TBRKON	6.2 7.2
5D	Disconnect Task from Interrupt	H.REXS,38	M.DISCON M_DISCON	6.2 7.2
5E	Exit from Message Receiver	H.REXS,39	M.XMSGR	6.2
5F	Parameter Task Activation	H.REXS,40	M.PTSK M_PTSK	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
60	Set User Abort Receiver Address	H.REXS,26	M.SUAR M_SUAR	6.2 7.2
61	Submit Job from Disc File	H.MONS,27	M.CDJS	6.4
62	Abort With Extended Message	H.REXS,28	M.BORT M_BORT	6.2 7.2
63	Load and Execute Interactive Debugger	H.REXS,29	M.DEBUG M_DEBUG	6.2 7.2
64	Get Task Number	H.REXS,32	M.ID M_ID M.MYID M_MYID	6.2 7.2 6.2 7.2
65	Get Address Limits	H.REXS,41	M.GADRL	6.2
66	Debug Link Service	H.REXS,42	N/A	6.3
	Debug Link Service - Base Mode	H.REXS,42	N/A	7.3
67	Get Dynamic Task Execution Space	H.REMM,10	M.GE	6.2
68	Free Dynamic Task Execution Space	H.REMM,11	M.FE	6.2
69	Get Dynamic Extended Data Space	H.REMM,8	M.GD	6.2
6A	Free Dynamic Extended Indexed Data Space	H.REMM,9	M.FD	6.2
6B	Receive Message Link Address	H.REXS,43	M.RCVR M_RCVR	6.2 7.2
6C	Send Message to Specified Task	H.REXS,44	M.SMSGR M_SMSGR	6.2 7.2
6D	Send Run Request to Specified Task	H.REXS,45	M.SRUNR M_SRUNR	6.2 7.2
6E	Break/Task Interrupt Link/Unlink	H.REXS,46	M.BRK M_BRK	6.2 7.2
6F	Activate Task Interrupt	H.REXS,47	M.INT M_INT	6.2 7.2

SVC 1,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
70	Exit from Task Interrupt Level	H.REXS,48	M.BRKXIT M.XBRKR	6.2 6.2
71	Share Memory with Another Task	H.ALOC,12	M.SHARE	6.4
72	Get Shared Memory	H.ALOC,13	M.INCL	6.4
73	Permanent File Log	H.MONS,33	M.LOG	6.4
74	User Name Specification	H.MONS,34	M.USER	6.4
75	Create Permanent File	H.FISE,12	M.CREATE	6.4
76	Change Temporary File to Permanent	H.FISE,13	M.PERM	6.4
77	Delete Permanent File or Non-SYSGEN Memory Partition	H.FISE,14	M.DELETE	6.4
78	Reserved			
79	Free Shared Memory	H.ALOC,14	M.EXCL	6.4
7A	Get Message Parameters	H.REXS,35	M.GMSGP M_GMSGP	6.2 7.2
7B	Get Run Parameters	H.REXS,36	M.GRUNP M_GRUNP	6.2 7.2
7C	Wait for Any No-wait Operation Complete, Message Interrupt, or Break Interrupt	H.REXS,37	M.ANYW M_ANYWAIT	6.2 7.2
7D	Exit Run Receiver	H.REXS,49	M.XRUNR	6.2
7E	Exit from Message End-action Routine	H.REXS,50	M.XMEA	6.2
7F	Exit from Run Request End-action Routine	H.REXS,51	M.XREA	6.2
80-FFF	Available for customer use			

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00-1F	Reserved			
20	Create Permanent File	H.VOMM,1	M.CPERM M_CREATEP	6.2 7.2
21	Create Temporary File	H.VOMM,2	M.TEMP M_CREATET	6.2 7.2
22	Create Memory Partition	H.VOMM,3	M.MEM M_MEM	6.2 7.2
23	Create Directory	H.VOMM,4	M.DIR M_DIR	6.2 7.2
24	Delete Resource	H.VOMM,5	M.DELR M_DELETER	6.2 7.2
25	Extend File	H.VOMM,6	M.EXTD M_EXTENDFILE	6.2 7.2
26	Truncate File	H.VOMM,7	M.TRNC M_TRUNCATE	6.2 7.2
27	Change Defaults	H.VOMM,8	M.DEFT M_DEFT	6.2 7.2
28	Change Temporary File to Permanent File	H.VOMM,9	M.TEMPER M_TEMPFILETOPERM	6.2 7.2
29	Log Resource or Directory	H.VOMM,10	M.LOGR M_LOGR	6.2 7.2
2A	Modify Descriptor	H.VOMM,11	M.MOD M_MOD	6.2 7.2
2B	Rewrite Descriptor	H.VOMM,12	M.REWRIT M_REWRIT	6.2 7.2
2C	Read Descriptor	H.VOMM,13	M.LOC M_READD	6.2 7.2
2D	Rename File	H.VOMM,14	M.RENAM M_RENAME	6.2 7.2
2E	Convert Pathname to Pathname Block	H.VOMM,15	M.PNAMB M_PNAMB	6.2 7.2
2F	Reconstruct Pathname	H.VOMM,16	M.PNAM M_CONSTRUCTPATH	6.2 7.2
30	Replace Permanent File	H.VOMM,23	M.REPLAC M_REPLACE	6.2 7.2
31	Modify Descriptor User Area	H.VOMM,26	M.MODU M_MODU	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
32	Rewrite Descriptor User Area	H.VOMM,27	M.REWRTU M_REWRTU	6.2 7.2
33	DBX Interface to H.PTRAC	N/A	N/A	N/A
34	Reserved for H.PTRAC			
35-3F	Reserved			
40	Include Memory Partition	H.REMM,12	M.INCLUDE	6.2
	Include Shared Image		M _INCLUDE	7.2
41	Exclude Memory Partition	H.REMM,14	M.EXCLUDE	6.2
	Exclude Shared Image		M _EXCLUDE	7.2
42	Open Resource	H.REMM,21	M.OPENR M_OPENR	6.2 7.2
43	Close Resource	H.REMM,22	M.CLOSER M_CLOSER	6.2 7.2
44	Set Exclusive Resource Lock	H.REMM,23	M.LOCK M_LOCK	6.2 7.2
45	Release Exclusive Resource Lock	H.REMM,24	M.UNLOCK M_UNLOCK	6.2 7.2
46	Set Synchronous Resource Lock	H.REMM,25	M.SETSYNC M_SETSYNC	6.2 7.2
47	Release Synchronous Resource Lock	H.REMM,26	M.UNSYNC M_UNSYNC	6.2 7.2
48	Resource Inquiry	H.REMM,27	M.INQUIRY M_INQUIRER	6.2 7.2
49	Mount Volume	H.REMM,17	M.MOUNT M_MOUNT	6.2 7.2
4A	Dismount Volume	H.REMM,19	M.DMOUNT M_DISMOUNT	6.2 7.2
4B	Get Memory in Byte Increments	H.REMM,28	M.MEMB M_GETMEMBYTES	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
4C	Free Memory in Byte Increments	H.REMM,29	M.MEMFRE M_FREEMEMBYTES	6.2 7.2
4D-4E	Reserved			
4F	Reserved			
50	Acquire Current Date/Time in ASCII Format	H.REXS,74	M.QATIM M_QATIM	6.2 7.2
	Acquire Current Date/Time in Binary Format	H.REXS,74	M.BTIM M_BTIM	6.2 7.2
	Acquire Current Date/Time in Byte Binary Format	H.REXS,74	M.BBTIM M_BBTIM	6.2 7.2
	Acquire System Date/Time in Any Format	H.REXS,74	M.GTIM M_GTIM	6.2 7.2
	Get Current Date and Time	H.REXS,74	M_GETTIME	7.2
51	Convert ASCII Date/Time to Byte Binary Format	H.REXS,75	M.CONABB M_CONABB	6.2 7.2
	Convert ASCII Date/Time to Standard Binary	H.REXS,75	M.CONASB M_CONASB	6.2 7.2
	Convert Binary Date/Time to ASCII Format	H.REXS,75	M.CONBAF M_CONBAF	6.2 7.2
	Convert Binary Date/Time to Byte Binary	H.REXS,75	M.CONBBY M_CONBBY	6.2 7.2
	Convert Byte Binary Date/Time to ASCII	H.REXS,75	M.CONBBA M_CONBBA	6.2 7.2
	Convert Byte Binary Date/Time to Binary	H.REXS,75	M.CONBYB M_CONBYB	6.2 7.2
	Convert System Date/Time Format	H.REXS,75	M.CTIM M_CTIM	6.2 7.2
	Convert Time	H.REXS,75	M_CONVERTTIME	7.2
52	Assign and Allocate Resource	H.REXS,21	M.ASSN M_ASSIGN	6.2 7.2

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
53	Deassign and Deallocate Resource	H.REXS,22	M.DASN M_DEASSIGN	6.2 7.2
54	Reformat RRS Entry	H.REXS,76	M.NEWRRS	6.2
55	Batch Job Entry	H.REXS,27	M.BATCH M_BATCH	6.2 7.2
56	Reserved for Interactive Debugger	H.REXS,30	N/A	N/A
57	Reinstate Privilege Mode to Privilege Task	H.REXS,78	M.PRIV M_PRIVMODE	6.2 7.2
58	Change Task to Unprivileged Mode	H.REXS,79	M.UPRIV M_UNPRIVMODE	6.2 7.2
59	Validate Address Range	H.REXS,33	M.VADDR M_VADDR	6.2 7.2
5A	Reserved			
5B	Set IPU Bias	H.REXS,82	M.IPUBS M_IPUBS	6.2 7.2
5C	Set Exception Handler	H.REXS,83	M_SETEXA	7.2
5D	Get Base Mode Task Address Limits	H.REXS,84	M_LIMITS	7.2
5E	Get Task Environment	H.REXS,85	M.ENVRMT M_ENVRMT	6.2 7.2
5F	Exit With Status	H.REXS,86	M_EXTSTS	7.2
60	Reserved			
61	Get Command Line	H.REXS,88	M.CMD M_CMD	6.2 7.2
62	Move Data to User Address	H.REXS,89	M.MOVE M_MOVE	6.2 7.2
63-6F	Reserved			
70	Get User Context	H.EXEC,41	M_GETCTX	7.2
71	Put User Context	H.EXEC,42	M_PUTCTX	7.2
72-74	Reserved for Symbolic Debugger/X32			
75	Reserved for MPX-32			

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
76	Allocate Shadow Memory	H.SHAD	N/A	N/A
77	Set Option Lower	H.TSM,13	M.SOPL M_SOPL	6.2 7.2
78	Reset Option Lower	H.TSM,14	M.ROPL M_ROPL	6.2 7.2
79	Set Exception Return Address	H.REXS,81	M_SETERA	7.2
7A	Get Terminal Function Definition	H.TSM,15	M.GETDEF M_GETDEF	6.2 7.2
7B	Get Address Limits	H.REXS,80	M.GADRL2	6.2
7C	Get Dynamic Extended Discontiguous Data Space	H.MEMM,9	M.GDD	6.2
7D	Get TSA Start Address	H.REXS,91	M.GTSAD M_GTSAD	6.2 7.2
7E	Physical Memory Read	H.REXS,93	M.OSREAD M_OSREAD	6.2 7.2
7F	Get Extended Memory Array	H.MEMM,14	N/A	6.3
	Get Extended Memory Array - Base Mode	H.MEMM,14	N/A	7.3
80-9F	Reserved for ACX-32			
A0-A3	Reserved for Swapper			
A4-A9	Reserved for Ada			
AA-AD	Reserved for Rapid File Allocation:		N/A	N/A
	Zero MDT	H.MDT,1		
	Locate/Read MDT Entry	H.MDT,2		
	Update/Create MDT Entry	H.MDT,3		
	Delete MDT Entry	H.MDT,4		
AE	TSM Procedure Call	H.TSM,17	M.TSMPC M_TSMPC	6.2 7.2
AF	Physical Memory Write	H.REXS,94	M.OSWRIT M_OSWRIT	6.2 7.2
B0-BE	Reserved for RMSS			
BF	Reserved			

SVC 2,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
C0	Task Option Doubleword Inquiry	H.REXS,95	M.PGOD M_OPTIONDWORD	6.2 7.2
C1-C7	Reserved			
N/A	Allocate File Space	H.VOMM,19	N/A	6.3
N/A	Allocate Resource Descriptor	H.VOMM,17	N/A	6.3
N/A	Create File Control Block	N/A N/A	M.DFCB M_CREATEFCB	5.9.1 7.2
N/A	Create Temporary File	H.VOMM,24	N/A	6.3
N/A	Deallocate File Space	H.VOMM,20	N/A	6.3
N/A	Deallocate Resource Descriptor	H.VOMM,18	N/A	6.3
N/A	Execute Channel Program File Control Block	N/A	M_CHANPROGFCB	7.2
N/A	Read/Write Authorization File	H.VOMM,25	N/A	6.3

SVC 5,X'*nn*'	Description	Module, E.P.	Macro	Volume I Ref.Manual Section
00	Suspend/Resume	N/A	M.SURE M_SURE	6.2 7.2

C MPX-32 Abort and Crash Codes

C.1 AC – Accounting

AC01	INSUFFICIENT SLO SPACE FOR ACCOUNTING LISTING

C.2 AD – Address Specification Trap Handler (H.IP0C)

AD01	ADDRESS SPECIFICATION ERROR OCCURRED WITHIN THE OPERATING SYSTEM
AD02	ADDRESS SPECIFICATION ERROR OCCURRED WITHIN THE CURRENT TASK
AD03	TRAP OCCURRED WHILE NO TASKS WERE IN ACTIVE STATE
AD04	TRAP OCCURRED WITHIN ANOTHER INTERRUPT TRAP ROUTINE

C.3 AL – Allocator (H.ALOC) (Compatibility Mode Only)

AL01–AL06	Reserved
AL07	THE COMBINED FILE ASSIGNMENTS FOR A TASK EXCEEDS NUMBER SPECIFIED. THE CATALOGED ASSIGNMENTS ARE COMBINED WITH THOSE DEFINED BY $ASSIGN STATEMENTS. SEE CATALOGER FILES DIRECTIVE AND RECATALOG IF NEEDED.
AL08	AN ASSIGNED PERMANENT FILE IS NONEXISTENT
AL09	AN ASSIGNED DEVICE IS NOT CONFIGURED IN THE SYSTEM. AN ASSIGNED DEVICE IS OFF-LINE.
AL10–AL11	Reserved
AL12	UNABLE TO LOAD PROGRAM BECAUSE OF I/O ERROR OR ADDRESSING INCONSISTENCIES IN LOAD MODULE PREAMBLE
AL13	AN UNRECOVERABLE I/O ERROR HAS OCCURRED DURING THE READ OF THE TASK PREAMBLE INTO THE TSA
AL14	Reserved
AL15	AN ASSIGNED DEVICE TYPE IS NOT CONFIGURED IN THE SYSTEM
AL16	A RESIDENT REQUEST HAS BEEN ISSUED FOR A TASK REQUIRING AN SLO, SBO, SGO OR SYC FILE. RESIDENT TASKS CANNOT USE SYSTEM FILES.
AL17–AL18	Reserved

AL19	A FILE CODE TO FILE CODE ASSIGNMENT (ASSIGN4) HAS BEEN MADE TO AN UNDEFINED FILE CODE. A FILE CODE MUST BE DEFINED BEFORE A SECOND FILE CODE CAN BE EQUATED BY AN ASSIGN4.
AL20	USER ATTEMPTED DEALLOCATION OF TSA
AL21	DESTROYED TASK MIDL WAS DETECTED WHILE ATTEMPTING TO ALLOCATE DYNAMIC EXECUTION SPACE
AL22	A SOFTWARE CHECKSUM ERROR HAS OCCURRED DURING TASK LOADING
AL23	AN INVALID USER NAME IS CATALOGED WITH THE TASK. THE USER NAME IS NOT CONTAINED IN THE M.KEY FILE OR A VALID KEY IS NOT SPECIFIED.
AL24	ACCESS TO AN ASSIGNED PERMANENT FILE IS BY PASSWORD ONLY, AND A VALID PASSWORD WAS NOT INCLUDED ON THE CATALOGED ASSIGNMENT OR JOB CONTROL STATEMENT ASSIGNMENT
AL25	UNDEFINED RESOURCE REQUIREMENT SUMMARY (RRS) TYPE (INTERNAL FORMAT OF AN ASSIGNMENT STATEMENT IS WRONG)
AL26	THE TASK HAS REQUESTED MORE BLOCKING BUFFERS THAN WERE SPECIFIED DURING CATALOG. SEE CATALOGER BUFFER DIRECTIVE AND RECATALOG IF NEEDED.
AL27	THERE ARE NO FREE ENTRIES IN SHARED MEMORY TABLE FOR GLOBAL, DATAPOOL, CSECT, OR OTHER SHARED AREAS
AL28	TASK IS ATTEMPTING TO SHARE AN UNDEFINED GLOBAL OR DATAPOOL MEMORY PARTITION
AL29	TASK IS ATTEMPTING TO EXCLUDE UNDEFINED MEMORY PARTITION
AL30	THE REQUESTED DEVICE IS ALREADY ASSIGNED TO THE REQUESTING TASK VIA ANOTHER FILE CODE. USE ASSIGN4 OR DEALLOCATE BEFORE REALLOCATING.
AL31	LOGICAL FILE CODE ALREADY ALLOCATED BY CALLER (E.G., A CARD READER MAY BE ASSIGNED TO LFC 'IN' AND A MAGNETIC TAPE CANNOT BE ASSIGNED TO THE SAME FILE CODE). USE ASSIGN4 OR DEALLOCATE BEFORE REALLOCATING.
AL32	DYNAMIC COMMON BLOCK MAY NOT BE ASSIGNED VIA ASSIGN1 DIRECTIVE
AL33	SHARED MEMORY DEFINITION CONFLICTS WITH CALLER'S ADDRESS SPACE
AL34	SHARED MEMORY PARTITION NOT DEFINED IN DIRECTORY
AL35	ATTEMPT TO SHARE A DIRECTORY ENTRY THAT IS NOT A MEMORY PARTITION

AL36	INVALID PASSWORD SPECIFIED FOR SHARED MEMORY PARTITION
AL37	ATTEMPT TO EXCLUDE UNDEFINED SHARED MEMORY PARTITION
AL38	ATTEMPT TO ACTIVATE A PRIVILEGED TASK BY UNAUTHORIZED OWNER
AL39	SHARED MEMORY ENTRY NOT FOUND
AL40	PARTITION DEFINITION NOT FOUND IN DIRECTORY
AL41	DIRECTORY DEFINITION NOT A DYNAMIC PARTITION
AL42	INVALID PASSWORD FOR A MEMORY PARTITION
AL43	TASK HAS ATTEMPTED TO ALLOCATE AN UNSHARED RESOURCE THAT WAS NOT AVAILABLE DURING TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL44	UNABLE TO RESUME 'SYSBUILD' TASK DURING INITIAL TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL45	UNABLE TO DEALLOCATE INPUT DEVICE AFTER DYNAMIC TASK ACTIVATION IN A MEMORY-ONLY ENVIRONMENT
AL46	TASK HAS ATTEMPTED TO SHARE MEMORY VIA A DYNAMIC MEMORY PARTITION IN A MEMORY-ONLY ENVIRONMENT
AL47	DYNAMIC MEMORY PARTITIONS CANNOT BE GREATER THAN 1 MEGABYTE
AL48	THE USER HAS ATTEMPTED TO EXCLUDE A SHARED PARTITION WHOSE ASSOCIATED MAP BLOCKS ARE NOT DESIGNATED AS BEING SHARED IN THE TASK'S TSA
AL49	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S TSA SPACE REQUIREMENTS
AL50	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S CSECT SPACE REQUIREMENTS, OR IF NO CSECT, LOAD MODULE IS TOO LARGE TO FIT IN USER'S ADDRESS SPACE
AL51	DESTROYED TASK MIDL DETECTED WHILE ATTEMPTING TO ALLOCATE SYSTEM BUFFER SPACE
AL52	AN ERROR CONDITION PERTAINING TO FILE SYSTEM STRUCTURES HAS OCCURRED. THIS ERROR IS NOT A FUNCTION OF THE COMPATIBILITY INTERFACE.
AL53	DESTROYED TASK MIDL WAS DETECTED WHILE ATTEMPTING TO ALLOCATE EXTENDED INDEXED DATA SPACE
AL54	INVALID COMPATIBLE RRS TYPE
AL55	ACCESS MODE IS NOT ALLOWED

C.4 AT – ANSI Labeled Tapes

AT01	INCORRECT OR NO RUN PARAMETERS RECEIVED
AT02	INCORRECT STATUS RETURNED FROM J.ATAPE RUN REQUEST
AT03	AN ERROR OCCURRED
AT04	I/O ERROR OCCURRED ON TAPE

C.5 AU – Auto-Start Trap Processor

AU01	TRAP OCCURRED ON AUTO-START

C.6 BT – Block Mode Timeout Trap

BT01	BLOCK MODE TIMEOUT TRAP

C.7 CM – Call Monitor Interrupt Processor (H.IP27 and H.IP0A)

CM01	CALL MONITOR INTERRUPT PROCESSOR CANNOT LOCATE THE 'CALM' INSTRUCTION
CM02	EXPECTED 'CALM' INSTRUCTION DOES NOT HAVE CALM (X'30') OPCODE
CM03	INVALID 'CALM' NUMBER
CM04	'CALM' NUMBER TOO LOW (OUT OF BOUNDS)
CM05	'CALM' NUMBER TOO BIG (OUT OF BOUNDS)

C.8 CP – Cache

CP01	CACHE PARITY ERROR OCCURRED WITHIN THE OPERATING SYSTEM
CP02	CACHE PARITY ERROR OCCURRED IN TASK BODY
CP03	TRAP OCCURRED WHILE NO TASKS WERE IN ACTIVE STATE
CP04	TRAP OCCURRED IN ANOTHER INTERRUPT TRAP ROUTINE

C.9 EX – Exit/Abort

EX01	AN ABORT HAS OCCURRED IN THE TASK EXIT SEQUENCE
EX02	AN ABORT HAS OCCURRED DURING THE TASK ABORT SEQUENCE AND HAS BEEN CHANGED TO A DELETE (KILL) TASK SEQUENCE
EX03	USER ATTEMPTED TO GO TO AN ANY WAIT STATE FROM AN END-ACTION ROUTINE

C.10 FS – File System (H.MONS)(Compatibility Mode Only)

FS01	UNRECOVERABLE I/O ERROR TO THE DIRECTORY
FS02	UNRECOVERABLE I/O ERROR TO FILE SPACE ALLOCATION MAP
FS03	ATTEMPT TO ADD A NEW FILE, BUT THE DIRECTORY IS FULL
FS04	A DISC ALLOCATION MAP CHECKSUM ERROR WAS DETECTED
FS05	ATTEMPT TO ALLOCATE DISC SPACE THAT IS ALREADY ALLOCATED
FS06	ATTEMPT TO DEALLOCATE DISC SPACE THAT IS NOT ALLOCATED
FS07	USER HAS CALLED AN ENTRY POINT IN H.FISE THAT NO LONGER EXISTS

C.11 HE – Online Help Facility

HE01	ABNORMAL TERMINATION WHILE TRANSLATING HELP FILES (HELPT)

C.12 HT – Halt Trap Processor (H.IPHT)

HT01	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION IN USER'S PROGRAM
HT02	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION IN AN INTERRUPT TRAP ROUTINE
HT03	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION WHEN NO TASKS WERE IN AN ACTIVE STATE
HT04	Reserved
HT05	AN ATTEMPT WAS MADE TO EXECUTE A HALT INSTRUCTION WHEN USER WAS UNMAPPED

C.13 IO – Input/Output Control Supervisor (H.IOCS)

IO01	Reserved
IO02	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ OR WRITE DATA INTO AN UNMAPPED ADDRESS
IO03	AN UNPRIVILEGED TASK IS ATTEMPTING TO READ DATA INTO PROTECTED MEMORY
IO04-IO05	Reserved
IO06	INVALID BLOCKING BUFFER CONTROL CELLS IN BLOCKED FILE ENCOUNTERED. PROBABLE CAUSES: (1) FILE IS IMPROPERLY BLOCKED, (2) BLOCKING BUFFER IS DESTROYED, OR (3) TRANSFER ERROR DURING FILE INPUT.
IO07	THE TASK HAS ATTEMPTED TO PERFORM AN OPERATION WHICH IS NOT VALID FOR THE DEVICE TO WHICH THE USER'S FILE IS ASSIGNED(E.G., A READ OPERATION SPECIFIED FOR A FILE ASSIGNED TO THE LINE PRINTER).
IO08	DEVICE ASSIGNMENT IS REQUIRED FOR AN UNPRIVILEGED TASK TO USE THIS SERVICE
IO09	ILLEGAL OPERATION ON THE SYC FILE
IO10-IO14	Reserved
IO15	A TASK HAS REQUESTED A TYPE OPERATION AND THE TYPE CONTROL PARAMETER BLOCK(TCPB) SPECIFIED INDICATES THAT AN OPERATION ASSOCIATED WITH THAT TCPB IS ALREADY IN PROGRESS
IO16	INVALID BLOCKING BUFFER CONTROL CELL(S) ENCOUNTERED DURING WRITE OF BLOCKED FILE. THIS ERROR IS USUALLY CAUSED BY A USER SPECIFIED BLOCKING BUFFER THAT HAS BEEN DESTROYED.
IO17	OPEN ATTEMPTED ON A FILE AND FPT HAS NO MATCHING FILE CODE. PROBABLE CAUSE: (1) BAD OR MISSING RRS IN PREAMBLE (2) LFC IN FCB HAS BEEN DESTROYED.
IO18	Reserved
IO19	AN ERROR HAS OCCURRED IN THE REMM CLOSE PROCEDURE
IO20	AN ERROR HAS OCCURRED IN THE REMM OPEN PROCEDURE
IO21	IOCS HAS ENCOUNTERED AN UNRECOVERABLE I/O ERROR IN ATTEMPTING TO PROCESS AN I/O REQUEST ON BEHALF OF A TASK
IO22	AN ILLEGAL IOCS ENTRY POINT HAS BEEN ENTERED BY A TASK
IO23	A H.VOMM DENIAL HAS OCCURRED IN READING THE RESOURCE DESCRIPTOR TO GET MORE SEGMENT DEFINITIONS

IO24	ILLEGAL ADDRESS, TRANSFER COUNT OR TRANSFER TYPE (I.E., IMPROPER BOUNDING FOR DATA TYPE) SPECIFIED IN THE FCB
IO25-IO27	Reserved
IO28	ILLEGAL OPERATION ATTEMPTED ON AN OUTPUT ACTIVE FILE OR DEVICE
IO29	Reserved
IO30	ILLEGAL OR UNEXPECTED VOLUME NUMBER OR REEL ID ENCOUNTERED ON MAGNETIC TAPE
IO31	Reserved
IO32	CALLING TASK HAS ATTEMPTED TO PERFORM A SECOND READ ON A '$' STATEMENT THROUGH THE SYC FILE
IO33	READ WITH BYTE GRANULARITY REQUEST MADE WITH NEGATIVE BYTE OFFSET
IO34	READ WITH BYTE GRANULARITY REQUEST MADE WITHOUT SETTING RANDOM ACCESS BIT IN FCB
IO35	READ WITH BYTE GRANULARITY REQUESTS ARE VALID FOR UNBLOCKED FILES ONLY
IO36-IO37	Reserved
IO38	WRITE ATTEMPTED ON UNIT OPENED IN READ-ONLY MODE. A READ-WRITE OPEN WILL BE FORCED TO READ-ONLY IF TASK HAS ONLY READ ACCESS TO UNIT.
IO39	Reserved
IO40	INVALID TRANSFER COUNT. TRANSFER COUNT TOO LARGE FOR TRANSFER TYPE, TRANSFER COUNT NOT AN EVEN MULTIPLE OF TRANSFER TYPE, OR DATA ADDRESS NOT BOUNDED FOR TRANSFER TYPE.
IO41	BLOCKING ERROR DURING NON-DEVICE ACCESS
IO42	BLOCKED DATA MANAGEMENT MODULE (H.BKDM) IS NOT CONFIGURED IN THE SYSTEM
IO43	INPUT/OUTPUT CONTROL LIST (IOCL) OR DATA ADDRESS NOT IN CONTIGUOUS 'E' MEMORY (GPMC DEVICES ONLY)
IO44	NON-DEVICE ACCESS I/O ERROR. THIS ERROR MAY BE THE RESULT OF CHANNEL/CONTROLLER INITIALIZATION FAILURE.
IO45	MULTIVOLUME MAGNETIC TAPE MODULE (H.MVMT) IS NOT CONFIGURED IN THE SYSTEM
IO46	Reserved
IO47	CLASS 'E' DEVICE TCW IS NOT IN CLASS 'E' MEMORY. THIS TYPE OF ERROR INDICATES A MAP FAILURE.

IO48–IO49	Reserved
IO50	AN UNPRIVILEGED USER ATTEMPTED TO EXECUTE A PHYSICAL CHANNEL PROGRAM
IO51	A 'TESTSTAR' COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO52	A LOGICAL CHANNEL WAS TOO LARGE TO BE MOVED TO MEMORY POOL
IO53	A 'TIC' COMMAND FOLLOWS A 'TIC' COMMAND IN A LOGICAL CHANNEL PROGRAM
IO54	A 'TIC' COMMAND ATTEMPTED TO TRANSFER TO AN ADDRESS WHICH IS NOT WORD BOUNDED
IO55	ILLEGAL ADDRESS IN LOGICAL IOCL. ADDRESS IS NOT IN USER'S LOGICAL ADDRESS SPACE.
IO56	A READ-BACKWARD COMMAND WAS USED IN A LOGICAL CHANNEL PROGRAM
IO57	ILLEGAL IOCL ADDRESS. IOCL MUST BE LOCATED IN THE FIRST 128K WORDS OF MEMORY.
IO58–IO60	Reserved
IO61	INVALID LFC IN FCB
IO62	ERROR OCCURRED ON IMPLICIT OPEN
IO63–IO76	Reserved
IO77	ATTEMPT TO USE DATA FLOW CONTROL (OTHER THAN WISM), THAT IS NOT SUPPORTED BY THE CURRENTLY INSTALLED CONTROLLER
IO78	ATTEMPT TO ISSUE AN EXECUTE CHANNEL PROGRAM TO A WRITE SUB-CHANNEL AND THE SUB-CHANNEL WAS NOT IN DUAL CHANNEL MODE
IO79	Reserved
IO80	ILLEGAL ACCESS MODE FOR VOLUME RESOURCE
IO81–IO97	Reserved
IO98	H.VOMM DENIAL HAS OCCURRED ON IOCS AUTOMATIC FILE EXTENSION REQUEST FOR THE LFC SPECIFIED IN THE ABORT MESSAGE
IO99	INTERNAL SYSTEM ERROR DETECTED AT THE ADDRESS RELATIVE TO IOCS WHICH IS SPECIFIED IN THE ABORT MESSAGE

C.14 IP – IPU

IP01	ABNORMAL TASK TERMINATION IN IPU

C.15 LD – Task Activation Loading (H.TAMM)

LD01	LOAD CODE SECTION ERROR
LD02	CODE SECTION CHECKSUM ERROR
LD03	BIAS CODE ERROR
LD04	CODE MATRIX CHECKSUM ERROR
LD05	LOAD DATA SECTION ERROR
LD06	DATA SECTION CHECKSUM ERROR
LD07	BIAS DATA ERROR
LD08	DATA MATRIX CHECKSUM ERROR
LD09	GCF R/O RELOCATION ERROR
LD10	GCF R/W RELOCATION ERROR

C.16 MC – Machine Check Trap

MC01	MACHINE CHECK TRAP

C.17 MF – Map Fault Trap

MF01	A MAP FAULT TRAP HAS OCCURRED. THIS IS THE RESULT OF A BAD MEMORY REFERENCE OUTSIDE OF THE USER'S ADDRESSABLE SPACE.

C.18 MM – Memory Disk

MM01	REQUEST FOR MEMORY DISC I/O TO A LOCATION OUTSIDE THE MEMORY DISC BOUNDARIES

C.19 MP – Memory Parity Trap (H.IP02)

MP01	MEMORY ERROR OCCURRED IN A TASK'S LOGICAL ADDRESS SPACE. THIS IS AN INTERNAL OR CPU FAILURE. RERUN TASK.
MP02	MEMORY ERROR OCCURRED IN ANOTHER INTERRUPT TRAP ROUTINE (NESTED TRAPS, CONTEXT LOST)
MP03	MEMORY ERROR OCCURRED WHILE NO TASKS WERE IN THE ACTIVE STATE
MP04	MEMORY ERROR OCCURRED IN A MAP BLOCK RESERVED FOR THE O/S
MP05	ERROR OCCURRED WHILE CURRENT TASK WAS IN THE UNMAPPED MODE

C.20 MS – System Services (H.MONS) (Compatibility Mode Only)

MS01	PERMANENT FILE ADDRESS INQUIRY SERVICE FOUND A NUMBER OF ALLOCATION UNITS IN THE UNIT DEFINITION TABLE THAT DO NOT CORRESPOND TO ANY KNOWN DISC.
MS02–MS08	Reserved
MS09	TASK HAS ATTEMPTED TO CONNECT A TASK TO AN INTERRUPT LEVEL NOT DEFINED FOR INDIRECTLY CONNECTED TASKS
MS10–MS11	Reserved
MS12	OVERLAY IS PASSWORD PROTECTED
MS13–MS15	Reserved
MS16	TASK HAS REQUESTED DYNAMIC ALLOCATION WITH AN INVALID FUNCTION CODE
MS17	FILE NAME CONTAINS CHARACTERS OUTSIDE RANGE OF X'20' TO X'5F', INCLUSIVELY
MS18–MS20	Reserved
MS21	MULTIVOLUME MAGNETIC TAPE ALLOCATION REQUEST MADE TO SCRATCH (SCRA) TAPE
MS22	MULTI-VOLUME MAGNETIC TAPE ALLOCATION REQUEST MADE ON SHARED TAPE DRIVE
MS23	TASK HAS ISSUED A 'MOUNT MESSAGE ONLY' ALLOCATION REQUEST TO A NON-ALLOCATED DRIVE OR TO A DEVICE WHICH IS NOT A MAGNETIC TAPE

MS24	TASK HAS SPECIFIED AN ILLEGAL VOLUME NUMBER (ZERO IF TAPE IS MULTIVOLUME, NONZERO IF TAPE IS SINGLE VOLUME)
MS25-MS27	Reserved
MS28	A PERMANENT FILE LOG HAS BEEN REQUESTED, BUT THE ADDRESS SPECIFIED FOR STORAGE OF THE DIRECTORY ENTRY IS NOT CONTAINED WITHIN THE CALLING TASK'S LOGICAL ADDRESS SPACE
MS29	Reserved
MS30	TASK HAS ATTEMPTED TO OBTAIN A PERMANENT FILE LOG IN A MEMORY-ONLY ENVIRONMENT
MS31	USER ATTEMPTED TO GO TO THE ANY-WAIT STATE FROM AN END-ACTION ROUTINE
MS32	Reserved
MS33	ALLOCATION ERROR IN RTM M.ALOC CALL
MS34-MS86	Reserved
MS87	NO DENIAL RETURN ADDRESS SPECIFIED ON CALM M.ALOC EMULATION

C.21 NM – Nonpresent Memory Trap

NM01	A NONPRESENT MEMORY TRAP ERROR CONDITION HAS OCCURRED.

C.22 OC – Operator Communications

OC01	THE OPERATOR HAS REQUESTED THAT THE TASK BE ABORTED

C.23 PT – Task Activation (J.TSM)

PT01	INVALID ATTEMPT TO MULTICOPY A UNIQUE TASK
PT02	FILE SPECIFIED IS NOT IN DIRECTORY
PT03	UNABLE TO ALLOCATE FILE
PT04	FILE IS NOT A VALID LOAD MODULE OR EXECUTABLE IMAGE
PT05	DQE IS NOT AVAILABLE
PT06	READ ERROR ON RESOURCE DESCRIPTOR

PT07	READ ERROR ON LOAD MODULE
PT08	INSUFFICIENT LOGICAL/PHYSICAL ADDRESS SPACE FOR TASK ACTIVATION
PT09	CALLING TASK IS UNPRIVILEGED
PT10	INVALID PRIORITY
PT11	INVALID SEND BUFFER ADDRESS OR SIZE
PT12	INVALID RETURN BUFFER ADDRESS OR SIZE
PT13	INVALID NO-WAIT MODE END ACTION ROUTINE ADDRESS
PT14	MEMORY POOL UNAVAILABLE
PT15	DESTINATION TASK RECEIVER QUEUE FULL
PT16	INVALID PSB ADDRESS
PT17	RRS LIST EXCEEDS 384 WORDS
PT18	INVALID RRS ENTRY IN PARAMETER BLOCK

C.24 PV – Privilege Violation Trap

PV01	PRIVILEGE VIOLATION TRAP

C.25 RC – Record Manager

RC01	LESS THAN ONE BLOCK ON READ
RC02	NOT A MULTIPLE NUMBER OF BLOCKS READ
RC03	NO MORE IOC'S AVAILABLE
RC04	ERROR CONDITION ON READ
RC05	PREMATURE END-OF-FILE
RC06	END-OF-MEDIUM ON OUTPUT FILE
RC07	WRITE ATTEMPTED ON UNOPENED FILE
RC08	USER RECORD SIZE TOO LARGE
RC09	READ NOT ALLOWED AFTER WRITE
RC10	ERROR ON WRITE
RC11	END-OF-MEDIUM ON OUTPUT FILE

RC12	INTERNAL FILE POSITION ERROR
RC13	RESOURCE CANNOT BE OPENED
RC14	INTERNAL FILE POSITION ERROR
RC15	INVALID BLOCKING BUFFER CELL

C.26 RE – Restart

RE01	RESTART IS INVALID IN BATCH OR COMMAND FILE MODE

C.27 RF – Rapid File Allocation

RF01	INVALID PATHNAME
RF02	PATHNAME CONSISTS OF VOLUME ONLY
RF03	VOLUME NOT MOUNTED
RF04	Reserved
RF05	FILE IS NOT A PERMANENT FILE
RF06	Reserved
RF07	RESOURCE DOES NOT EXIST
RF08	RESOURCE NAME IN USE
RF09	Reserved
RF10	MDT ENTRY UNAVAILABLE
RF11–RF14	Reserved
RF15	VOLUME MUST BE MOUNTED PUBLIC
RF16–RF59	Reserved
RF60	INVALID MODE
RF61–RF98	Reserved
RF99	WARNING, INPUT ERRORS ENCOUNTERED, CHECK SLO OUTPUT

C.28 RM – Resource Management (H.REMM)

RM01	UNABLE TO LOCATE RESOURCE
RM02	ACCESS MODE NOT ALLOWED

RM03	TOO MANY ASSIGNMENTS
RM04	BLOCKING BUFFER SPACE NOT AVAILABLE OR INVALID BUFFER ADDRESS
RM05	SHARED MEMORY TABLE (SMT) ENTRY NOT FOUND
RM06	TOO MANY MOUNT REQUESTS
RM07	STATIC ASSIGN TO DYNAMIC COMMON
RM08	UNRECOVERABLE I/O ERROR
RM09	INVALID USAGE SPECIFICATION
RM10	INVALID PARAMETER ADDRESS
RM11	INVALID RESOURCE REQUIREMENT SUMMARY (RRS) ENTRY
RM12	INVALID LFC TO LFC ASSIGNMENT
RM13	DEVICE NOT IN SYSTEM OR OFF-LINE
RM14	RESOURCE ALREADY ALLOCATED BY TASK
RM15	INVALID SYC/SGO ASSIGNMENT
RM16	COMMON CONFLICTS WITH TASK ADDRESS SPACE
RM17	DUPLICATE LFC ASSIGNMENT
RM18	INVALID DEVICE SPECIFICATION
RM19	INVALID RESOURCE ID (RID)
RM20	VOLUME UNASSIGNED OR ACCESS NOT ALLOWED
RM21	UNABLE TO MOUNT. J.MOUNT RUN REQUEST FAILED
RM22	RESOURCE MARKED FOR DELETION
RM23	ASSIGNED DEVICE IS MARKED OFF-LINE
RM24	UNABLE TO LOCATE MOUNTED VOLUME TABLE(MVT) ENTRY
RM25	RANDOM ACCESS NOT ALLOWED
RM26	ATTEMPT TO WRITE ON SYC
RM27	RESOURCE ALREADY OPENED IN DIFFERENT MODE
RM28	INVALID ACCESS SPECIFICATION AT OPEN
RM29	INVALID FILE CONTROL BLOCK(FCB) ADDRESS OR UNASSIGNED LFC IN FCB
RM30	INVALID ALLOCATION INDEX

RM31	RESOURCE NOT OPEN
RM32	LOCK NOT OWNED BY THIS TASK
RM33	RESOURCE IS NOT ALLOCATED IN A SHARABLE MODE
RM34	SYSTEM ADMINISTRATOR ATTRIBUTE IS REQUIRED TO MOUNT A PUBLIC VOLUME
RM35	RESOURCE IS NOT A SHARED IMAGE
RM36	PHYSICAL MEMORY ALREADY ALLOCATED
RM37	ATTEMPT TO ALLOCATE NONPRESENT PHYSICAL MEMORY
RM38	TIME OUT WAITING FOR RESOURCE
RM39	UNABLE TO PERFORM WRITE BACK
RM40	INVALID LOAD MODULE
RM41	INVALID PHYSICAL ADDRESS SPECIFIED
RM42	USER REQUESTED ABORT OF MOUNT PROCESS
RM43	USER REQUESTED HOLD ON MOUNT PROCESS
RM44	WRITEBACK REQUESTED AND SHARED IMAGE HAS NO WRITEBACK SECTION
RM45	LOADING ERROR DURING INCLUSION OF READ ONLY SECTION OF SHARED IMAGE
RM46	UNABLE TO OBTAIN RESOURCE DESCRIPTOR LOCK (MULTIPORT ONLY)
RM47	LOADING ERROR DURING INCLUSION OF READ/WRITE SECTION OF SHARED IMAGE
RM48	INCOMPATIBLE LOAD ADDRESSES FOR SHARED IMAGE
RM49	TASK HAS REQUESTED EXCESSIVE NUMBER OF MULTICOPIED SHARED IMAGES WITH NO READ ONLY SECTION
RM50	RESOURCE IS LOCKED BY ANOTHER TASK
RM51	SHAREABLE RESOURCE IS ALLOCATED BY ANOTHER TASK IN AN INCOMPATIBLE ACCESS MODE
RM52	VOLUME SPACE IS NOT AVAILABLE
RM53	ASSIGNED DEVICE IS NOT AVAILABLE
RM54	UNABLE TO ALLOCATE RESOURCE FOR SPECIFIED USAGE
RM55	ALLOCATED RESOURCE TABLE (ART) SPACE IS NOT AVAILABLE

RM56	TASK REQUIRES SHADOW MEMORY AND NONE IS CONFIGURED
RM57	VOLUME IS NOT AVAILABLE FOR MOUNT WITH REQUESTED USAGE
RM58	SHARED MEMORY TABLE (SMT) SPACE IS NOT AVAILABLE
RM59	MOUNTED VOLUME TABLE (MVT) SPACE IS NOT AVAILABLE
RM60	RESOURCE DESCRIPTOR SPACE DEFINITION CONFLICT
RM61	UNABLE TO LOCATE OR RETRIEVE RESOURCE DESCRIPTOR
RM62	INVALID OPTION IN CNP
RM63	SEGMENTED TASK SUPPORT NOT PRESENT.
RM64	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S TASK SERVICE AREA(TSA) SPACE REQUIREMENTS
RM65	THE TASK'S DSECT SPACE REQUIREMENTS OVERLAP THE TASK'S CSECT SPACE REQUIREMENTS, OR IF NO CSECT, LOAD MODULE IS TOO LARGE TO FIT IN USER'S ADDRESS SPACE
RM66	SOFTWARE CHECKSUM. ERROR MAY BE FIXED BY RECATALOGING.
RM67	EXCESSIVE MEMORY REQUEST
RM68	EXCESSIVE VOLUME SPACE REQUESTED
RM69	INVALID USERNAME SPECIFIED
RM70	INVALID PRIVILEGED ACTIVATION
RM71	Reserved
RM72	UNABLE TO RESUME SYSINIT ON TAPE ACTIVATION
RM73	FILE OVERLAP HAS OCCURRED. PLEASE CHECK THE SYSTEM CONSOLE
RM74	LOADING ERROR
RM75	INVALID WORK VOLUME/DIRECTORY
RM76	USER ATTEMPTED DEALLOCATION OF TSA
RM77	A TASK HAS DESTROYED THE ALLOCATION LINKAGES IN ITS DYNAMIC EXPANSION SPACE
RM78	UNABLE TO LOAD TASK DEBUGGER WITH TASK
RM79	INVALID CALLER NOTIFICATION PACKET (CNP) ADDRESS
RM80	SHARED IMAGE VERSION LEVEL IS NOT COMPATIBLE WITH EXECUTABLE IMAGE

RM81	INVALID ACTIVATION OF A BASE MODE TASK ON A SYSTEM CONFIGURED FOR NON-BASE TASK EXECUTION.
RM82	INVALID ACTIVATION OF AN ADA TASK ON A SYSTEM CONFIGURED WITHOUT ADA SUPPORT.
RM83	INSUFFICIENT LOGICAL ADDRESS SPACE TO ACTIVATE TASK
RM84	INVALID LOGICAL POSITION FOR EXTENDED MPX
RM85	PTRACE DEBUG REQUESTED AND H.PTRAC NOT CONFIGURED
RM86	CANNOT DISMOUNT THE SYSTEM VOLUME.
RM87	PUBLIC VOLUME DISMOUNT DENIED DUE TO COMPATIBLE MODE PUBLIC DISMOUNT OPTION SET FOR THIS SYSTEM.
RM88	PUBLIC DISMOUNT DENIED. SYSTEM ADMINISTRATOR ATTRIBUTE REQUIRED FOR THIS OPERATION.
RM89	PUBLIC DISMOUNT DENIED DUE TO MISSING OPTION FOR PUBLIC VOLUME IN THE DISMOUNT REQUEST
RM90	GCL LOADMODULE OR SHIM CANNOT BE RELOCATABLE
RM91	UNABLE TO ACCESS VOLUME DUE TO PENDING PHYSICAL DISMOUNT.
RM92	READ ONLY OR READ WRITE LOAD ADDRESS IS INVALID
RM93	UNABLE TO PERFORM PHYSICAL MOUNT DUE TO SYSTEM SHUTDOWN IN PROGRESS.
RM94	J.MOUNT ATTEMPTED TO MOUNT AN UNFORMATTED DISC VOLUME.
RM95	AN UNBIASED TASK REQUIRES SHADOW MEMORY ON A SYSTEM WITH NO OVERLAPPING CPU/IPU SHADOW REGION
RM96	A BIASED TASK REQUIRES SHADOW MEMORY THAT DOES NOT EXIST ON THE SPECIFIED PROCESSOR
RM97	Reserved
RM98	THE TASK REQUIRES MORE SHADOW MEMORY THAN EXISTS

C.29 RX – Resident Executive Services (H.REXS)

RX01	Reserved
RX02	INVALID FUNCTION CODE SPECIFIED FOR REQUEST TO CREATE A TIMER ENTRY. VALID CODES ARE ACP (1), RSP OR RST (2), STB (3), RSB (4) AND RQI (5).
RX03	TASK ATTEMPTED TO SET/RESET A BIT OUTSIDE OF A STATIC PARTITION OR THE OPERATING SYSTEM.

RX04	THE REQUESTING TASK IS UNPRIVILEGED OR HAS ATTEMPTED TO CREATE A TIMER ENTRY TO REQUEST AN INTERRUPT WITH A PRIORITY LEVEL OUTSIDE THE RANGE OF X'12' TO X'7F', INCLUSIVELY
RX05	INVALID FUNCTION CODE HAS BEEN SPECIFIED FOR REQUEST TO SET USER STATUS WORD
RX06	UNPRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY LEVEL, OR A PRIVILEGED TASK ATTEMPTED TO RESET A TASK PRIORITY TO A LEVEL OUTSIDE THE RANGE OF 1 TO 64, INCLUSIVELY
RX07	CANNOT LOAD OVERLAY SEGMENT DUE TO SOFTWARE CHECKSUM OR DATA ERROR
RX08	OVERLAY IS NOT IN THE DIRECTORY
RX09	Reserved
RX10	OVERLAY HAS AN INVALID PREAMBLE
RX11	AN UNRECOVERABLE I/O ERROR HAS OCCURRED DURING OVERLAY LOADING
RX12	Reserved
RX13	FUNCTION CODE SUPPLIED TO A DATE/TIME SERVICE IS OUT OF RANGE
RX14	DESTINATION BUFFER ADDRESS IS INVALID OR PROTECTED
RX15	ATTEMPT TO SET EXCEPTION RETURN ADDRESS WHEN ARITHMETIC EXCEPTION NOT IN PROGRESS
RX16-RX24	Reserved
RX25	OPERATOR HAS ABORTED TASK IN RESPONSE TO MOUNT MESSAGE
RX26-RX28	Reserved
RX29	TASK HAS ATTEMPTED TO LOAD THE INTERACTIVE TASK DEBUGGER OVERLAY IN A MEMORY-ONLY ENVIRONMENT
RX30-RX31	Reserved
RX32	INVALID DQE ADDRESS
RX33	OVERLAY LINKAGES HAVE BEEN DESTROYED BY LOADING A LARGER OVERLAY
RX34	TASK HAS MADE A BREAK RECEIVER EXIT CALL WHILE NO BREAK IS ACTIVE
RX35	Reserved

RX36	STATUS IN REGISTER ZERO IS NOT A ZERO OR A VALID ABORT CODE
RX37-RX85	Reserved
RX86	TASK HAS MADE AN END ACTION ROUTINE EXIT WHILE END ACTION WAS NOT ACTIVE
RX87	Reserved
RX88	RESERVED FOR DEBUG LINK SERVICE
RX89	AN UNPRIVILEGED TASK HAS ATTEMPTED TO REESTABLISH AN ABORT RECEIVER (OTHER THAN M.IOEX)
RX90	TASK HAS MADE A RUN REQUEST END ACTION ROUTINE EXIT WHILE THE RUN REQUEST INTERRUPT WAS NOT ACTIVE
RX91	TASK HAS ATTEMPTED NORMAL EXIT WITH A TASK INTERRUPT STILL ACTIVE
RX92	TASK HAS ATTEMPTED NORMAL EXIT WITH MESSAGES IN ITS RECEIVER QUEUE
RX93	AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT
RX94	AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING MESSAGE EXIT
RX95	TASK HAS MADE A MESSAGE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE
RX96	AN INVALID RECEIVER EXIT BLOCK (RXB) ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT
RX97	AN INVALID RECEIVER EXIT BLOCK (RXB) RETURN BUFFER ADDRESS WAS ENCOUNTERED DURING RUN RECEIVER EXIT
RX98	TASK HAS MADE A RUN RECEIVER EXIT WHILE THE RUN RECEIVER INTERRUPT WAS NOT ACTIVE
RX99	TASK HAS MADE A MESSAGE END-ACTION ROUTINE EXIT WHILE THE MESSAGE INTERRUPT WAS NOT ACTIVE

C.30 SB – System Binary Output

SB01	AN I/O ERROR HAS BEEN ENCOUNTERED ON THE DEVICE ASSIGNED AS THE SYSTEM BINARY (PUNCHED) OUTPUT DEVICE
SB02	THE SYSTEM OUTPUT PROGRAM HAS ENCOUNTERED AN UNRECOVERABLE I/O ERROR IN ATTEMPTING TO READ A PUNCHED OUTPUT FILE FROM DISC
SB03	DENIAL OF FILE CODE TO FILE CODE ALLOCATION FOR J.SOUT2 INDICATES LOSS OF SYSTEM INTEGRITY

SB04	SYSTEM BINARY OUTPUT ABORTED BY OPERATOR
SB05	NO TIMER ENTRY FOR SYSTEM BINARY OUTPUT (SYSTEM FAULT)
SB06	FIVE ECHO CHECK ERRORS DETECTED WHILE ATTEMPTING TO PUNCH A SINGLE CARD

C.31 SC – System Check Trap Processor

SC01	SYSTEM CHECK TRAP OCCURRED AT AN ADDRESS LOCATED WITHIN THE OPERATING SYSTEM
SC02	SYSTEM CHECK TRAP OCCURRED WITHIN THE CURRENT TASK'S SPACE
SC03	SYSTEM CHECK TRAP OCCURRED AT A TIME WHEN THERE WERE NO TASKS CURRENTLY BEING EXECUTED (C.PRNO EQUALS ZERO)
SC04	SYSTEM CHECK TRAP OCCURRED WITHIN ANOTHER TRAP (C.GINT DOES NOT EQUAL '1')

C.32 SD – SCSI Disk

SD00	NO ADDITIONAL SENSE INFORMATION
SD01	NO INDEX/SECTOR SIGNAL
SD02	NO SEEK COMPLETE
SD03	WRITE FAULT
SD04	DRIVE NOT READY
SD05	DRIVE NOT SELECTED
SD06	NO TRACK ZERO FOUND
SD07	MULTIPLE DRIVES SELECTED
SD08	LOGICAL UNIT COMMUNICATIONS FAILURE
SD09	TRACK FOLLOWING ERROR
SD10–SD15	Reserved
SD16	ID CRC OR ECC ERROR
SD17	UNRECOVERED READ ERROR OF DATA BLOCKS
SD18	NO ADDRESS MARK FOUND IN ID FIELD

SD19	NO ADDRESS MARK FOUND IN DATA FIELD
SD20	NO RECORD FOUND
SD21	SEEK POSITIONING ERROR
SD22	DATA SYNCHRONIZATION MARK ERROR
SD23	RECOVERED READ DATA WITH TARGET'S READ RETRIES (NOT WITH ECC)
SD24	RECOVERED READ DATA WITH TARGET'S ECC CORRECTION (NOT WITH RETRIES)
SD25	DEFECT LIST ERROR
SD26	PARAMETER OVERRUN
SD27	SYNCHRONOUS TRANSFER ERROR
SD28	PRIMARY DEFECT LIST NOT FOUND
SD29	COMPARE ERROR
SD30	RECOVERED ID WITH TARGET'S ECC CORRECTION
SD31	Reserved
SD32	INVALID COMMAND OPERATION CODE
SD33	ILLEGAL LOGICAL BLOCK ADDRESS. ADDRESS GREATER THAN THE LBA RETURNED BY THE READ CAPACITY DATA WITH PMI BIT NOT SET IN CDB
SD34	ILLEGAL FUNCTION FOR DEVICE TYPE
SD35	Reserved
SD36	ILLEGAL FIELD IN CDB
SD37	INVALID LUN
SD38	INVALID FIELD IN PARAMETER LIST
SD39	WRITE PROTECTED
SD40	MEDIUM CHANGE
SD41	POWER ON OR RESET OR BUS DEVICE RESET OCCURRED
SD42	MODE SELECT PARAMETERS CHANGED
SD43-SD47	Reserved
SD48	IMCOMPATIBLE CARTRIDGE

SD49	MEDIUM FORMAT CORRUPTED
SD50	NO DEFECT SPARE LOCATION AVAILABLE
SD51-SD63	Reserved
SD64	RAM FAILURE
SD65	DATA PATH DIAGNOSTIC FAILURE
SD66	POWER ON DIAGNOSTIC FAILURE
SD67	MESSAGE REJECT ERROR
SD68	INTERNAL CONTROLLER ERROR
SD69	SELECT/RESELECT FAILED
SD70	UNSUCCESSFUL SOFT RESET
SD71	SCSI INTERFACE PARITY ERROR
SD72	INITIATOR DETECTED ERROR
SD73	INAPPROPRIATE/ILLEGAL MESSAGE

C.33 SG – System Generator (SYSGEN)

SG01	INVALID LOADER FUNCTION CODE IN BINARY OBJECT MODULE FROM THE SYSTEM RESIDENT MODULE (OBJ) FILE
SG02	INVALID BINARY RECORD READ FROM SYSTEM RESIDENT MODULE (OBJ) FILE (BYTE 0 MUST BE X'FF' OR X'DF')
SG03	SEQUENCE ERROR IN MODULE BEING READ FROM TEMPORARY FILE
SG04	CHECKSUM ERROR IN MODULE BEING READ FROM TEMPORARY FILE
SG05	UNABLE TO FIND CDT AND/OR UDT FOR I/O MODULE LOAD
SG06	UNABLE TO OBTAIN ADDITIONAL MEMORY REQUIRED FOR RESIDENT SYSTEM IMAGE MODULE LOADING
SG07	UNABLE TO OBTAIN MEMORY REQUIRED FOR RESIDENT SYSTEM IMAGE CONSTRUCTION
SG08	NON-RELOCATABLE BYTE STRING ENCOUNTERED IN BINARY MODULE BEING PROCESSED FROM TEMPORARY FILE
SG09	UNABLE TO ALLOCATE TEMPORARY FILE SPACE
SG10	OVERRUN OF SYSGEN ADDRESS SPACE BY SYSTEM BEING GENERATED. PROBABLE ERRONEOUS SIZE SPECIFICATION IN PATCH OR POOL DIRECTIVE.

SG11	SEQUENCE ERROR WHILE READING OBJECT MODULE FROM FILE ASSIGNED TO 'OBJ'
SG12	CHECKSUM ERROR WHILE READING OBJECT MODULE FROM FILE ASSIGNED TO 'OBJ'
SG13	UNABLE TO ALLOCATE DISC SPACE FOR SYMTAB FILE. POSSIBLE CAUSES ARE INSUFFICIENT DISC SPACE OR ACCESS RIGHTS DENIAL.
SG14	UNABLE TO ALLOCATE DISC SPACE FOR SYSTEM IMAGE FILE. POSSIBLE CAUSES ARE INSUFFICIENT DISC SPACE, ACCESS RIGHTS DENIAL, OR ATTEMPTING TO SYSGEN OVER CURRENT DEFAULT IMAGE.
SG15	MAXIMUM NUMBER (240) OF SYMBOL TABLE/PATCH FILE ENTRIES EXCEEDED
SG16	MISSING SYSTEM OR SYMTAB DIRECTIVE
SG17	INVALID IPU INTERVAL TIMER PRIORITY. MUST NOT BE BETWEEN X'78' AND X'7F'.
SG18	MAXIMUM SIZE OF 88K FOR TARGET SYSTEM HAS BEEN EXCEEDED
SG19	ATTEMPT TO DEFINE INTERRUPT VECTORING ROUTINE AS SYSTEM REENTRANT. ONLY DEVICE HANDLERS MAY BE SYSTEM REENTRANT.
SG20	UNABLE TO FIND "LINK" DEVICE IN UDT
SG21	INSUFFICIENT ROOM IN MEMORY POOL FOR DOWNLOAD FILE LIST
SG22	Reserved
SG23	SHARE DIRECTIVE SPECIFIED WITHOUT ENOUGH SMT ENTRIES. ENTRIES MUST EXCEED OR BE EQUAL TO THE NUMBER OF PARTITIONS PLUS MEMORY DISCS.
SG24	ATTEMPT TO DEFINE PARTITION STARTING MAPBLOCK NUMBER IN OPERATING SYSTEM AREA
SG25	ATTEMPT TO DEFINE PARTITION STARTING MAPBLOCK NUMBER IN NON-CONFIGURED PHYSICAL MEMORY
SG26	ATTEMPT TO USE A MODULE INCOMPATIBLE WITH THE TARGET MACHINE TYPE. THE OFFENDING MODULE NAME IS THE LAST ENTRY ON THE LISTING FOLLOWED BY THREE ASTERISKS (***).
SG27	THE DEVICE SPECIFIED IN EITHER THE SWAPDEV, SID, LOD OR POD DIRECTIVE IS NOT INCLUDED IN THE CONFIGURATION BEING BUILT
SG28	THE NULL DEVICE SPECIFICATION WHICH IS REQUIRED TO BE INCLUDED IN EVERY CONFIGURATION IS MISSING

SG29	SYSINIT OBJECT MODULE MISSING ON SYSGEN OBJECT INPUT FILE (OBJ). IT MUST BE THE LAST MODULE.
SG30	THE FILE ASSIGNED TO FILE CODE OBJ DOES NOT CONTAIN VALID OBJECT CODE
SG31	THE GENERATED IMAGE CONTAINS UNSATISFIED EXTERNAL REFERENCES. SEE THE SLO OUTPUT FOR MORE DETAILS. THIS IS NOT A FATAL ABORT AND THE SYSTEM IMAGE IS PRODUCED.
SG32	ONE OR MORE REQUESTED OBJECT MODULES COULD NOT BE LOCATED ON THE INPUT OBJECT FILE. SEE THE SLO OUTPUT FOR MORE DETAILS. THIS IS NOT A FATAL ABORT AND THE SYSTEM IMAGE IS PRODUCED.
SG33	EVENT TRACE HAS BEEN ENABLED WITH NO MEMORY PARTITION RESERVED FROM X'78000' TO X'80000'
SG34	Reserved
SG35	INSUFFICIENT MEMORY POOL FOR STATIC PARTITION
SG36	UNMAPPED DEBUG MODULE (H.DBUG2) IS MISSING ON SYSGEN OBJECT INPUT FILE. IT MUST BE THE LAST MODULE IF THE SYSTEM DEBUGGER IS TO BE CONFIGURED.
SG37	COMMUNICATION REGION + DSECT + ADAPTIVE REGION EXCEEDS 16KW
SG38	MPX EXTENDED CODE AREA EXTENDS PAST LOGICAL LIMIT
SG39	INVALID MPX EXTENDED CODE AREA LOGICAL MAP START
SG40	DIRECTIVE ERRORS ENCOUNTERED. IMAGE PRODUCED.
SG41	H.IPPF COULD NOT BE LOCATED ON THE INPUT OBJECT FILE. MODULE IS NECESSARY FOR DEMAND PAGE.
SG42–SG97	Reserved
SG98	ERROR ENCOUNTERED DURING OBJECT PROCESSING PRECEDED BY MESSAGE DESCRIBING THE ERROR CONDITION
SG99	DIRECTIVE ERRORS ENCOUNTERED

C.34 SH – Shadow Memory (J.SHAD)

SH01	J.SHAD ABORTED. SEE OUTPUT (UT IF INTERACTIVE OR SLO IF BATCH), FOR ACTUAL ERROR DESCRIPTION(S).

C.35 SN – System Input Task (J.SSIN)

SN00	INVALID RUN REQUEST PARAMETERS

C.36 SS – Sort/Merge (FSORT2)

SS01	CTL NOT ALLOCATED
SS02	HEADER DIRECTIVE MISSING
SS03	CONTROL FILE EMPTY
SS04	DIRECTIVE CODE NOT VALID
SS05–SS06	Reserved
SS07	OUTPUT FILE CODE (OUT) NOT ALLOCATED
SS08	RECORD LENGTH NOT DIVISIBLE INTO INPUT PHYSICAL RECORD LENGTH
SS09	RECORD LENGTH EXCEEDS INPUT PHYSICAL RECORD LENGTH
SS10	INPUT RECORD LENGTH EXCEEDS MAXIMUM ALLOWED (4095)
SS11	RECORD LENGTH NOT DIVISIBLE INTO OUTPUT PHYSICAL RECORD LENGTH
SS12	RECORD LENGTH EXCEEDS OUTPUT BLOCK LENGTH
SS13	OUTPUT PHYSICAL RECORD LENGTH EXCEEDS MAXIMUM ALLOWED (4095)
SS14	..1 PRESENT BUT NOT A DISC FILE
SS15	..2 PRESENT BUT NOT A DISC FILE
SS16	COMPARISON INDICATOR NOT VALID
SS17	Reserved
SS18	WK1 HAS BEEN ALLOCATED BY THE USER
SS19	WK2 HAS BEEN ALLOCATED BY THE USER
SS20	FIELD DIRECTIVE ERROR: STARTING POSITION IS GREATER THAN FIELD ENDING POSITION
SS21	FIELD DIRECTIVE ERROR: STARTING POSITION EXCEEDS RECORD LENGTH
SS22	FIELD DIRECTIVE ERROR: ENDING POSITION EXCEEDS LOGICAL RECORD LENGTH
SS23–SS27	Reserved
SS28	INAPPROPRIATE COMBINATION OF TOURNAMENT PARAMETERS EXCEEDS MEMORY POOL LIMITS
SS29	DISC SPACE CANNOT BE ALLOCATED FOR WORK FILE 1

SS30	DISC SPACE CANNOT BE ALLOCATED FOR WORK FILE 2
SS31	FILE TO FILE ALLOCATION FOR WORKFILE HAS FAILED
SS32	SORT BUFFER TOO SMALL
SS33-SS39	Reserved
SS40	INPUT FILES ARE EMPTY: NO RECORD INPUT OR SORTED
SS41	WK1 OR WK2 FILES TOO SMALL
SS42	MERGE ONLY SELECTED BUT NO MERGE FILES (MG1-MG8) ARE ASSIGNED
SS43-SS47	Reserved
SS48	SORT ATTEMPTED WITHOUT GOOD CALL TO SORT:HDR
SS49-SS57	Reserved
SS58	INAPPROPRIATE COMBINATION OF BUFFER PARAMETERS DETECTED DURING OUTPUT PHASE
SS59	END OF MEDIUM DETECTED ON THE OUT FILE
SS60-SS68	Reserved
SS69	COMPARE TABLE TYPE DESTROYED: SORT PROBLEM
SS70-SS97	Reserved
SS98	ERROR OPENING FILE LO
SS99	ERROR OPENING FILE OUT

C.37 ST – System Output Task (J.SOUT)

ST01	UNRECOVERABLE WRITE ERROR TO DESTINATION DEVICE
ST02	UNABLE TO PERFORM ALLOCATION OF SEPARATOR FILE CODE
ST03	UNABLE TO ISSUE MAGNETIC TAPE MOUNT MESSAGE VIA ALLOCATION SERVICE

Whenever a system output task aborts, the task may be restarted with the OPCOM REPRINT or REPUNCH commands.

C.38 SV – SVC Trap Processor (H.IP06)

SV01	UNPRIVILEGED TASK ATTEMPTING TO USE M.CALL
SV02	INVALID SVC NUMBER

SV03	UNPRIVILEGED TASK ATTEMPTING TO USE A 'PRIVILEGED-ONLY' SERVICE
SV04	INVALID SVC TYPE
SV05	UNPRIVILEGED TASK ATTEMPTING TO USE M.RTRN
SV06	INVALID MODULE NUMBER OR ENTRY POINT
SV07	ATTEMPTING TO USE A SVC WHICH IS INVALID FOR BASE REGISTER OPERATIONS
SV08	SVC 0, 1 OR 2 ATTEMPTED THAT WOULD RESULT IN A TSA STACK OVERFLOW (I.E. T.REGP GREATER THAN T.LASTP)
SV09	ATTEMPT TO USE A COMPATIBLE MODE SERVICE WITH NOCMS SPECIFIED IN SYSGEN

C.39 SW – Swap Scheduler Task (J.SWAPR)

SW01	I/O ERROR ON INSWAP OR OUTSWAP
SW02	EOM DETECTED ON SWAP FILE
SW03	CAN NOT CREATE SWAP FILE SPACE DIRECTORY IN MEMORY POOL
SW04	SWAP FILE SPACE DIRECTORY IS FULL
SW05	TASK HAS REQUESTED INSWAP BUT WAS NEVER OUTSWAPPED

C.40 SX – System Output Executive (J.SOEX)

SX01	INVALID RUN REQUEST HEADCELL COUNT
SX02	LOAD MODULE J.SOUT DOES NOT EXIST

C.41 SY – System Initialization (SYSINIT)

SY01	SYSTEM HALT OCCURRED DURING SYSINIT PHASE ONE PROCESSING
SY02	SYSTEM HALT DUE TO MEMORY PARITY ERROR BEING DETECTED IN THE OPERATING SYSTEM

C.42 TD – Terminal Type Set/Reset Utility (J.TSET)

TD01	ATTEMPTED TO RUN J.TSET IN BATCH MODE
TD02	J.TSET WAS UNABLE TO OPEN UT FOR PROCESSING

C.43 TS – Terminal Support

TS01	USER REQUESTED REMOVAL FROM A BREAK REQUEST
TS02	USER REQUESTED REMOVAL FROM A RESOURCE WAIT STATE QUEUE
TS03	TASK RUNNING FROM SPECIFIED TERMINAL WAS ABORTED WHEN THE TERMINAL DISCONNECTED
TS04	REMOVAL OF A JOB WAS REQUESTED

C.44 UI – Undefined Instruction Trap

UI01	UNDEFINED INSTRUCTION TRAP
UI02	UNEXPECTED DEBUGX32 BREAKPOINT FOUND AND DEBUGX32 NOT ATTACHED

C.45 VF – Volume Formatter (J.VFMT)

VF01	ERROR HAS OCCURRED. SEE SLO FILE FOR EXPLANATION.
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

C.46 VM – Volume Management Module (H.VOMM)

In some cases, H.VOMM displays H.REMM abort conditions. If a user calls an H.VOMM service which in turn calls an H.REMM service for processing and an abort condition occurs within the H.REMM processing, the abort condition is returned to H.VOMM which displays it to the user in the format `10`*xx* where *xx* is the specific H.REMM abort condition. For example, abort condition `1026` indicates H.REMM error 26 has occurred. The TSM $ERR command can be used to determine the reason for the error, i.e., `$ERR RM26`.

VM01	INVALID PATHNAME
VM02	PATHNAME CONSISTS OF VOLUME ONLY

VM03	VOLUME NOT MOUNTED
VM04	DIRECTORY DOES NOT EXIST
VM05	DIRECTORY NAME IN USE
VM06	DIRECTORY CREATION NOT ALLOWED AT SPECIFIED LEVEL
VM07	RESOURCE DOES NOT EXIST
VM08	RESOURCE ALREADY EXISTS
VM09	RESOURCE DESCRIPTOR UNAVAILABLE
VM10	DIRECTORY ENTRY UNAVAILABLE
VM11	REQUIRED FILE SPACE UNAVAILABLE
VM12	UNRECOVERABLE I/O ERROR READING DMAP
VM13	UNRECOVERABLE I/O ERROR WRITING DMAP
VM14	UNRECOVERABLE I/O ERROR READING RESOURCE DESCRIPTOR
VM15	UNRECOVERABLE I/O ERROR WRITING RESOURCE DESCRIPTOR
VM16	UNRECOVERABLE I/O ERROR READING SMAP
VM17	UNRECOVERABLE I/O ERROR WRITING SMAP
VM18	UNRECOVERABLE I/O ERROR READING DIRECTORY
VM19	UNRECOVERABLE I/O ERROR WRITING DIRECTORY
VM20	PROJECTGROUP NAME OR KEY INVALID
VM21	Reserved
VM22	INVALID FILE CONTROL BLOCK(FCB) OR LFC
VM23	PARAMETER ADDRESS SPECIFICATION ERROR
VM24	RESOURCE DESCRIPTOR NOT CURRENTLY ALLOCATED
VM25	PATHNAME BLOCK OVERFLOW
VM26	FILE SPACE NOT CURRENTLY ALLOCATED
VM27	'CHANGE DEFAULTS' NOT ALLOWED
VM28	RESOURCE CANNOT BE ACCESSED IN REQUESTED MODE OR DEFAULT SYSTEM IMAGE FILE CANNOT BE DELETED
VM29	OPERATION NOT ALLOWED ON THIS RESOURCE TYPE (RESOURCE IS NOT CORRECT TYPE)
VM30	REQUIRED PARAMETER WAS NOT SPECIFIED

VM31	FILE EXTENSION DENIED. SEGMENT DEFINITION AREA FULL.
VM32	FILE EXTENSION DENIED. FILE WOULD EXCEED MAXIMUM SIZE ALLOWED.
VM33	I/O ERROR OCCURRED WHEN RESOURCE WAS ZEROED
VM34	REPLACEMENT FILE CANNOT BE ALLOCATED
VM35	INVALID DIRECTORY ENTRY
VM36	DIRECTORY AND FILE ARE NOT ON THE SAME VOLUME
VM37	AN UNIMPLEMENTED ENTRY POINT HAS BEEN CALLED
VM38	REPLACEMENT FILE IS ALLOCATED BY ANOTHER TASK AND BIT 0 IN THE CNP OPTION FIELD IS NOT SET, OR FILE IS ALLOCATED BY OTHER CPU IN MULTI-PORT ENVIRONMENT
VM39	OUT OF SYSTEM SPACE
VM40	CANNOT ALLOCATE FAT/FPT WHEN CREATING A TEMPORARY FILE
VM41	DEALLOCATE ERROR IN ZEROING FILE
VM42	RESOURCE DESCRIPTOR DESTROYED OR THE RESOURCE DESCRIPTOR AND THE DIRECTORY ENTRY LINKAGE HAS BEEN DESTROYED
VM43	INVALID RESOURCE SPECIFICATION
VM44	INTERNAL LOGIC ERROR FROM RESOURCE MANAGEMENT MODULE (H.REMM). ABORT TASK, TRY A DIFFERENT TASK AND IF IT FAILS, REBOOT SYSTEM.
VM45	ATTEMPTED TO MODIFY MORE THAN ONE RESOURCE DESCRIPTOR AT THE SAME TIME OR ATTEMPTED TO REWRITE A RESOURCE DESCRIPTOR PRIOR TO MODIFYING IT
VM46	RESOURCE DESCRIPTOR IS LOCKED BY ANOTHER CPU (MULTI-PORT ONLY)
VM47	DIRECTORY CONTAINS ACTIVE ENTRIES AND CANNOT BE DELETED
VM48	A RESOURCE DESCRIPTOR'S LINK COUNT IS ZERO
VM49	ATTEMPTING TO DELETE A PERMANENT RESOURCE WITHOUT SPECIFYING A PATHNAME OR PATHNAME BLOCK VECTOR
VM50	RESOURCE DESCRIPTOR CONTAINS UNEXPECTED RESOURCE DESCRIPTOR TYPE
VM51	DIRECTORY ENTRY DELETED BUT FAILED TO RELEASE FILE SPACE

VM52	AN ATTEMPT WAS MADE TO DEALLOCATE FREE SPACE OR TO ALLOCATE SPACE THAT IS CURRENTLY ALLOCATED ON A VOLUME OTHER THAN SYSTEM DISC
VM53	THE FILE SPACE CREATED IS LESS THAN THE SPACE REQUESTED
VM54–VM98	Reserved
VM99	AN ATTEMPT WAS MADE TO DEALLOCATE FREE SPACE OR TO ALLOCATE SPACE THAT IS CURRENTLY ALLOCATED ON THE SYSTEM VOLUME

C.47 VO – Volume Manager (VOLMGR)

VO01	ERROR HAS OCCURRED. SEE SLO FILE FOR EXPLANATION.
VO02	OPEN FAILURE ON AUDIT TRAIL DEVICE/FILE
VO03	EOF/EOM ON AUDIT TRAIL DEVICE/FILE
VO04	I/O ERROR ON AUDIT TRAIL DEVICE/FILE
VO05	Reserved
VO06	I/O ERROR ON THE TAPE DURING SAVE OPERATION. TAPE HAS BEEN BACKSPACED TO THE END OF THE LAST SAVED FILE. ALL FILES ON THE IMAGE PRIOR TO THE TAPE I/O ERROR ARE SAVED ON THE TAPE.

C.48 Crash Codes

When system crash occurs as a result of a trap handler entry, the CPU halts with the registers containing the following information:

Register	Contents	
0	PSD Word 0 (when trap generated)	
1	PSD Word 1 (when trap generated)	
2	Real address of instruction causing trap	
3	Instruction causing trap	
4	CPU status word (from trap handler)	
5	Crash code:	
	MP01=X'4D503031'	(See H.IP02 Codes)
	NM01=X'4E4D3031'	(Nonpresent Memory - H.IP03)
	UI01=X'55493031'	(Undefined Instruction - H.IP04)
	PV01=X'50563031'	(Privilege Violation - H.IP05)
	MC01=X'4D433031'	(Machine Check - H.IP07)
	SC01=X'53433031'	(System Check - H.IP08)
	MF01=X'4D463031'	(Map Fault - H.IP09)
	CP01=X'43503031'	(Cache Parity Error - H.IP10) 32/67, 32/87 and 32/97
	SW01=X'53573031'	(See SWAPR codes)
6	Real address of register save block	
7	C'TRAP'=X'54524150'	

For further description, see Volume I, Chapter 2.

D Numerical Information

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5

87D13C01

2^n	n	2^{-n}
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 380 086 994 171 142 578 125

87D13D01

E Powers of Integers

E.1 Powers of Sixteen in Decimal

16^n	n	16^{-n}
1	0	0.10000 00000 00000 00000 x 10
16	1	0.62500 00000 00000 00000 x 10^{-1}
256	2	0.39062 50000 00000 00000 x 10^{-2}
4 096	3	0.24414 06250 00000 00000 x 10^{-3}
65 536	4	0.15258 78906 25000 00000 x 10^{-4}
1 048 576	5	0.95367 43164 06250 00000 x 10^{-6}
16 777 216	6	0.59604 64477 53906 25000 x 10^{-7}
268 435 456	7	0.37252 90298 46191 40625 x 10^{-8}
4 294 967 296	8	0.23283 06436 53869 62891 x 10^{-9}
68 719 476 736	9	0.14551 91522 83668 51807 x 10^{-10}
1 099 511 627 776	10	0.90949 47017 72928 23792 x 10^{-11}
17 592 186 044 416	11	0.56843 41886 08080 14870 x 10^{-13}
281 474 976 710 656	12	0.35527 13678 80050 09294 x 10^{-14}
4 503 599 627 370 496	13	0.22204 46049 25031 30808 x 10^{-15}
72 057 594 037 927 936	14	0.13877 78780 78144 56755 x 10^{-16}
1 152 921 504 606 846 976	15	0.86736 17379 88403 54721 x 10^{-18}

E.2 Powers of Ten in Hexadecimal

			10^n	n	10^{-n}					
			1	0	1.0000	0000	0000	0000		
			A	1	0.1999	9999	9999	999A		
			64	2	0.28F5	C28F	5C28	F5C3	x	16^{-1}
			3E8	3	0.4189	374B	C6A7	EF9E	x	16^{-2}
			2710	4	0.68DB	8BAC	710C	B296	x	16^{-3}
		1	86A0	5	0.A7C5	AC47	1B47	8423	x	16^{-4}
		F	4240	6	0.10C6	F7A0	B5ED	8D37	x	16^{-4}
		98	9680	7	0.1AD7	F29A	BCAF	4858	x	16^{-5}
		5F5	E100	8	0.2AF3	1DC4	6118	73BF	x	16^{-6}
		3B9A	CA00	9	0.44B8	2FA0	9B5A	52CC	x	16^{-7}
	2	540B	E400	10	0.6DF3	7F67	5EF6	EADF	x	16^{-8}
	17	4876	E800	11	0.AFEB	FF0B	CB24	AAFF	x	16^{-9}
	E8	D4A5	1000	12	0.1197	9981	2DEA	1119	x	16^{-9}
	918	4E72	A000	13	0.1C25	C268	4976	81C2	x	16^{-10}
	5AF3	107A	4000	14	0.2D09	370D	4257	3604	x	16^{-11}
3	8D7E	A4C6	8000	15	0.480E	BE7B	9D58	566D	x	16^{-12}
23	86F2	6FC1	0000	16	0.734A	CA5F	6226	F0AE	x	16^{-13}
163	4578	5D8A	0000	17	0.B877	AA32	36A4	B449	x	16^{-14}
DF0	B6B3	A764	0000	18	0.1272	5DD1	D243	ABA1	x	16^{-14}
8AC7	2304	89E8	0000	19	0.1D83	C94F	B6D2	AC35	x	16^{-15}

F ASCII Interchange Code Set

Row	Col	0	1	2	3	4	5	6	7
Bit Positions									
4	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	1	1	1	1
6	2	0	0	1	1	0	0	1	1
7	3	0	1	0	1	0	1	0	1
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No Punch	0 0	@ 8-4	P 11-7	` 8-1	p 12-11-7
0001	1	SOH 12-9-1	DC1 11-9-1	! 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8
1010	A	LF 0-9-5	SUB 9-8-7	* 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9
1011	B	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0
1100	C	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	¦ 12-11
1101	D	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-5	m 12-11-4	} 11-0
1110	E	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	^ 11-8-7	n 12-11-5	~ 11-0-1
1111	F	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7

87D13B01

Some positions in the ASCII code chart may have different graphic representation on various devices as:

ASCII	IBM 029
!	|
[	¢
]	!
^	>

Control Characters:

NUL	-	Null
SOH	-	Start of Heading (CC)
STX	-	Start of Text (CC)
ETX	-	End of Text (CC)
EOT	-	End of Transmission (CC)
ENQ	-	Enquiry (CC)
ACK	-	Acknowledge (CC)
BEL	-	Bell (audible or attention signal)
BS	-	Backspace (FE)
HT	-	Horizontal Tabulation (punch card skip) (FE)
LF	-	Line Feed (FE)
VT	-	Vertical Tabulation (FE)
FF	-	Form Feed (FE)
CR	-	Carriage Return (FE)
SO	-	Shift Out
SI	-	Shift In
DLE	-	Data Link Escape (CC)
DC1	-	Device Control 1
DC2	-	Device Control 2
DC3	-	Device Control 3
DC4	-	Device Control 4 (stop)
NAK	-	Negative Acknowledge (CC)
SYN	-	Synchronous Idle (CC)
ETB	-	End of Transmission Block (CC)
CAN	-	Cancel
EM	-	End of Medium
SS	-	Start of Special Sequence
ESC	-	Escape
FS	-	File Separator (IS)
GS	-	Group Separator (IS)
RS	-	Record Separator (IS)
US	-	Unit Separator (IS)
DEL	-	Delete
SP	-	Space (normally nonprinting)
(CC)	-	Communication Control
(FE)	-	Format Effector
(IS)	-	Information Separator

87D13C02

G IOP/MFP Panel Mode Commands

AS	Clear address stop
AS=*xxxxxxxx*	Set address stop at address *xxxxxxxx*
BAS	Read base registers
BAS*n*=*xxxxxxxx*	Write base register *n* (0-7) with *xxxxxxxx*
CLE	Clear memory
CRMD=*xxxxxxxxxxxx* =*xxxxxxxxxxxx*	Load CRAM with *xxxxxxxxxxxx* Load CRAM with data and increment address
CS	Read control switches
CS=*xxxxxxxx*	Set control switches to *xxxxxxxx*
EA	Read effective address
EXEC	Execute CRAM
GPR	Read general purpose registers
GPR*n*=*xxxxxxxx*	Write general purpose register *n* (0-7) with *xxxxxxxx*
HALT	Halt
IPL	IPL from default address
IPL=*xxxx*	IPL from channel/subaddress *xxxx*
IS	Clear instruction stop
IS=*xxxxxxxx*	Set instruction stop at address *xxxxxxxx*
MA=*xxxxxx* <ret>	Read physical memory address location *xxxxxx* Increment and read memory address
MAV=*xxxxxx* <ret>	Read virtual memory address location *xxxxxx* Increment and read memory address
MD=*xxxxxxxx* =*xxxxxxxx* <ret>	Write memory data *xxxxxxxx* into last location addressed Increment and write memory data *xxxxxxxx* Increment and write previous data
MSGE	Message between primary and secondary panels (IOP only)
OVR	Toggle clock override

PC=*xxxxxx*	Load program counter with address *xxxxxx*
PRIP	Set primary panel (master; IOP only)
PSD	Read program status doubleword (1 and 2)
PSD=*xxxxxxxx*	Write program status word (2) with *xxxxxxxx*
PSW	Read program status word (1)
PSW=*xxxxxxxx*	Write program status word (1) with *xxxxxxxx*
RS	Clear read operand stop
RS=*xxxxxxxx*	Set read operand stop at address *xxxxxxxx*
RST	Reset
RUN	Run
SECP	Set secondary panel (master and slave; IOP only)
STEP <ret>	Instruction step Continuation of instruction step
WS	Clear write operand stop
WS=*xxxxxxxx*	Set write operand stop at address *xxxxxxxx*
@@C	Enter console mode
@@P	Enter panel mode
(LF)	Repeat command

Notes:

1. Press the return key (<ret>) after each command.
2. LOCK ON and LOCK OFF are not supported by the CRT panel.

Console Mode

To change from panel mode to console mode, enter @@C<ret>.

Upon receipt of the <ret> following the @@C command, the firmware moves the cursor on the CRT to the extreme left margin of the next line.

To return to the panel mode, enter @@P<ret>. When the panel mode is selected, // is the prompt.

H Standard Date and Time Formats

H.1 Description

With the advent of the new MPX-32 file system, proper maintenance of the system date and time becomes more important than ever before as all file system resources will be time stamped to aid in management. It is vital the date and time be kept in a manner that is at once useful in this application and also convenient to convert into other formats that the user might require.

System date and time are kept in standard binary format. This format consists of two words: the first word contains the date and the second word contains the time. The date is maintained as the number of days since January 1, 1960 and the time is maintained as the binary count of system time units since midnight, adjusted to 100 microsecond granularity.

For the convenience of the user, monitor service calls are provided to convert the date and time between any of three standard formats. These are:

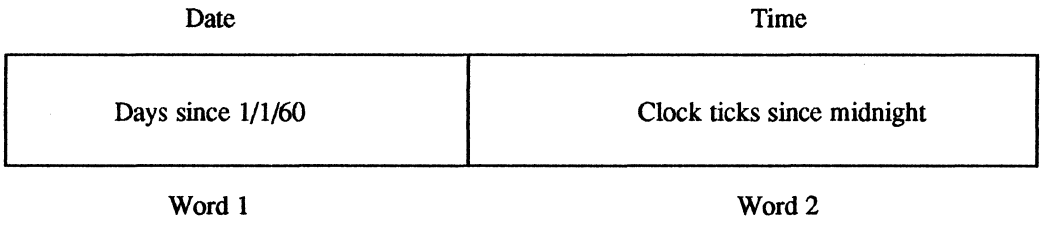
1. Binary Format (described above)
2. Byte Binary Format
3. ASCII Format (sometimes referred to as quad ASCII format)

Byte binary format time consists of two words: the first word contains date information and the second word contains time information. In byte binary format, the date is kept as four distinct values instead of one. Byte 0 of the date word is the binary century, byte 1 is the binary year in that century, byte 2 is the binary month and byte 3 the binary day of the month. Time is kept in a similar manner with byte 0 being the hour, byte 1 the minute, byte 2 the second, and byte 3 the number of clock ticks.

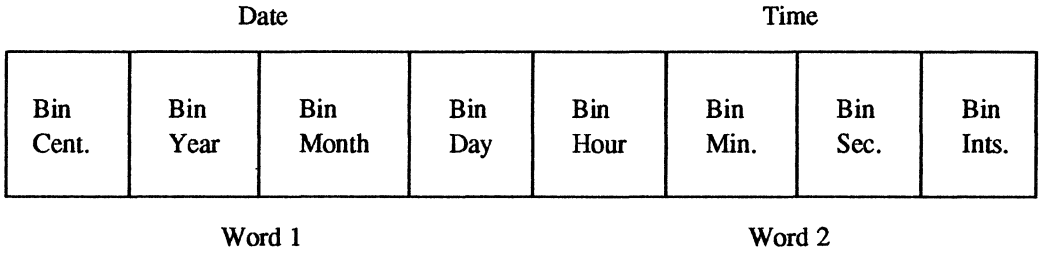
ASCII format consists of four words of information. The first two words contain the ASCII century, year, month, and day in successive halfwords. The second two words contain the hour, minutes, seconds, and clock ticks in a similar fashion. In ASCII format, use of a 120-hertz clock can cause truncation of the clock tick fields, allowing for only two ASCII digits.

H.2 Date/Time Standard Formats

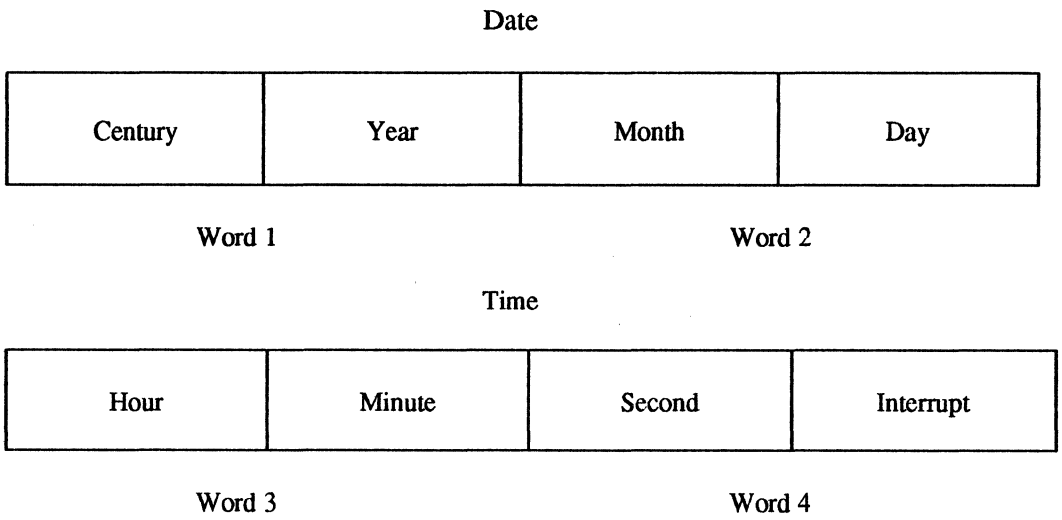
Binary



Byte Binary



Quad ASCII



I Compressed Source Format

Compressed source files are blocked files that consist of 120 byte records. The last record may be less than 120 bytes and has a data type code of 9F. The structure of a compressed record is described below.

Each record contains 6 control bytes:

1 byte	data type code, BF (9F indicates last record)
1 byte	byte count, number of data bytes in record
2 bytes	checksum, halfword sum of data bytes
2 bytes	sequence number, record sequence number starting at zero

Data is recorded as follows:

1 byte	blank count, number of blanks before data
1 byte	data count, number of data bytes
n-bytes	actual ASCII data
.	
.	(this sequence is repeated until the end of a line is reached)
.	
1 byte	EOL character, FF

J Map Block Address Assignments

Map Block # Decimal/Hex	Page # Decimal/Hex	Address Range Hexadecimal
00/00	00/00	00000 - 01FFF
01/01	04/04	02000 - 03FFF
02/02	08/08	04000 - 05FFF
03/03	12/0C	06000 - 07FFF
04/04	16/10	08000 - 09FFF
05/05	20/14	0A000 - 0BFFF
06/06	24/18	0C000 - 0DFFF
07/07	28/1C	0E000 - 0FFFF
08/08	32/20	10000 - 11FFF
09/09	36/24	12000 - 13FFF
10/0A	40/28	14000 - 15FFF
11/0B	44/2C	16000 - 17FFF
12/0C	48/30	18000 - 19FFF
13/0D	52/34	1A000 - 1BFFF
14/0E	56/38	1C000 - 1DFFF
15/0F	60/3C	1E000 - 1FFFF
16/10	64/40	20000 - 21FFF
17/11	68/44	22000 - 23FFF
18/12	72/48	24000 - 25FFF
19/13	76/4C	26000 - 27FFF
20/14	80/50	28000 - 29FFF
21/15	84/54	2A000 - 2BFFF
22/16	88/58	2C000 - 2DFFF
23/17	92/5C	2E000 - 2FFFF
24/18	96/60	30000 - 31FFF
25/19	100/64	32000 - 33FFF
26/1A	104/68	34000 - 35FFF
27/1B	108/6C	36000 - 37FFF
28/1C	112/70	38000 - 39FFF
29/1D	116/74	3A000 - 3BFFF
30/1E	120/78	3C000 - 3DFFF
31/1F	124/7C	3E000 - 3FFFF
32/20	128/80	40000 - 41FFF
33/21	132/84	42000 - 43FFF
34/22	136/88	44000 - 45FFF
35/23	140/8C	46000 - 47FFF
36/24	144/90	48000 - 49FFF
37/25	148/94	4A000 - 4BFFF

Map Block # Decimal/Hex	Page # Decimal/Hex	Address Range Hexadecimal
38/26	152/98	4C000 - 4DFFF
39/27	156/9C	4E000 - 4FFFF
40/28	160/A0	50000 - 51FFF
41/29	164/A4	52000 - 53FFF
42/2A	168/A8	54000 - 55FFF
43/2B	172/AC	56000 - 57FFF
44/2C	176/B0	58000 - 59FFF
45/2D	180/B4	5A000 - 5BFFF
46/2E	184/B8	5C000 - 5DFFF
47/2F	188/BC	5E000 - 5FFFF
48/30	192/C0	60000 - 61FFF
49/31	196/C4	62000 - 63FFF
50/32	200/C8	64000 - 65FFF
51/33	204/CC	66000 - 67FFF
52/34	208/D0	68000 - 69FFF
53/35	212/D4	6A000 - 6BFFF
54/36	216/D8	6C000 - 6DFFF
55/37	220/DC	6E000 - 6FFFF
56/38	224/E0	70000 - 71FFF
57/39	228/E4	72000 - 73FFF
58/3A	232/E8	74000 - 75FFF
59/3B	236/EC	76000 - 77FFF
60/3C	240/F0	78000 - 79FFF
61/3D	244/F4	7A000 - 7BFFF
62/3E	248/F8	7C000 - 7DFFF
63/3F	252/FC	7E000 - 7FFFF
Extended Memory 128KW to 256KW - 1B		
64/40	256/100	80000 - FFFFF
Extended Memory 256KW to 384KW - 1B		
128/80	512/200	100000 - 17FFFF
Extended Memory 384KW to 512KW - 1B		
192/C0	768/300	180000 - 1FFFFF
Extended Memory 512KW to 1024KW - 1B		
256/100	1024/400	200000 - 3FFFFF
Extended Memory 1024KW to 2048KW - 1B		
512/200	2048/800	400000 - 7FFFFF
Extended Memory 2048KW to 4096KW - 1B		
1024/400	4096/1000	800000 - FFFFFF

K Control Switches

While rebooting the system, various initialization processes can be inhibited or enabled by setting the appropriate control switches. The switch assignments are:

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 Reference Manual, Volume III, Chapter 9, Entry Conditions).
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 R2 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	Inhibits initialization of the memory descriptor table (MDT).
12	For RMSS: inhibits booting of nodes while J.BOOT executes.

The control switches can be accessed by the 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 3, the following must be entered on the IOP/MFP console:

```
ENTER DATE AND TIME: @@P
//CS=10000000    Terminal Initialization Inhibited
//@@C
<CR>
INVALID DATE FORMAT=MM/DD/XX
ENTER DATE AND TIME:
```

Refer to the CONCEPT 32/2000 Operations manual for instructions for setting control switches on the Amiga console.

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., online restart and IPL).

L Data Structures

L.1 Introduction

This appendix contains some of the more frequently used data structures. Below is a list of those structures.

Caller Notification Packet (CNP)
Controller Definition Table (CDT)
Dispatch Queue Entry (DQE)
File Control Block (FCB), 16 Word
File Control Block (FCB), 8 Word
File Control Block (FCB), High Speed Data
File Pointer Table (FPT)
Parameter Task Activation Block (PTASK)
TSM Procedure Call Block (PCB)
Pathname Blocks (PNB)
Post Program-Controlled Interrupt Notification Packet (PPCI)
Parameter Receive Block (PRB)
Parameter Send Block (PSB)
Resource Create Block (RCB)
Resource Identifiers (RID)
Resource Logging Block (RLB)
Resource Requirement Summary (RRS) Entries
Receiver Exit Block (RXB)
Type Control Parameter Block (TCPB)
Unit Definition Table (UDT)

L.2 Caller Notification Packet (CNP)

The caller notification packet (CNP) is the mechanism used by the Resource Management Module (H.REMM) and the Volume Management Module (H.VOMM) for handling abnormal conditions that may result during resource requests. All or part of this structure can be used by a particular service being called. The CNP must be on a word boundary.

	0 7	8 15	16 23	24 31
Word 0	Time-out value (CP.TIMO)			
1	Abnormal return address (CP.ABRET)			
2	Option field (CP.OPTS). See Note 1.		Status field (CP.STAT). See Note 2.	
3-4	Reserved (See Note 3.)			
5	Automatic open FCB address (CP.FCBA)			

Notes:

1. A bit sequence and/or value used to provide additional information that can be necessary to fully define the calling sequence for a particular service.
2. A right-justified numeric value identifying the return status for this call.
3. Refer to the individual system service description in the MPX-32 Reference Manual Volume I for interpretation of these words.

L.3 Controller Definition Table (CDT)

The controller definition table (CDT) is a system resident structure used to identify information required by handlers and the I/O processor for a specific controller. The CDT is built by the SYSGEN process, one for each controller configured on the system. The CDT identifies devices (UDTs) associated with the controller, the handler address associated with the controller, and defines other pertinent controller information.

	0 7	8 15	16 23	24 31
Word 0	String forward address (CDT.FIOQ)			
1	String backward address (CDT.BIOQ)			
2	Link priority (CDT.LPRI). See Note 1.	Number of entries in list (CDT.IOCT). See Note 2.	Class (CDT.CLAS). See Note 3.	Flags (CDT.FLG2). See Note 4.
3	CDT index (CDT.INDX)		Device type code (CDT.DTC) See Note 5.	Interrupt priority level (CDT.IPL)
4	Number units on controller (CDT.NUOC)	Number requests outstanding (CDT.IORO)	Channel number (CDT.CHAN)	Subaddress of first device (CDT.SUBA)
5	Program number if reserved (CDT.PNRC)	Interrupt handler address (CDT.SIHA) or controller information block (CDT.CIF)		
6	Flags (CDT.FLGS). See Note 6.	UDT address of first device on controller (CDT.UDTA)		
7	I/O status (CDT.IOST). See Note 7.	TI address (CDT.TIAD) or SI address if extended I/O (CDT.SIAD)		
8	UDT address unit 0* (CDT.UT0)			
9-23	UDT address unit 1* (CDT.UT1) through UDT address unit 15* (CDT.UTF)			

*Initialized by SYSGEN

Notes:

1. Always zero (head cell)
2. Number of entries in list (zero if none)

3. Values in CDT.CLAS are assigned as follows:

Value	Meaning
X'0D'	TCW type with extended addressing capability
X'0E'	TCW type
X'0F'	extended I/O

4. Bits in CDT.FLG2 are assigned as follows:

Bit	Meaning if Set
0	SCSI device (CDT.SCSI)
1-7	reserved for future use

5. For example, 01 for any disk, 04 for any tape, etc. Valid device type codes are listed in Appendix A.

6. Bits in CDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	extended I/O device (CDT.FCLS)
1	I/O outstanding (set by handler, reset by IOCS) (CDT.IOU1)
2	GPMC device (CDT.GPMC)
3	initialization (INC) needs to be performed for this controller (CDT.FINT)
4	D-class (CDT.XGPM)
5	used only when IOQs are linked to the CDT. Set when SIO is accepted by the controller. Reset when IOQ is unlinked from the CDT or when I/O is reported complete to IOCS in the case of operator intervention type errors (CDT.IOU5).
6	IOP controller (CDT.IOP)
7	controller malfunction (CDT.MALF)

7. Bits in CDT.IOST are assigned as follows:

Bit	Meaning if Set
0	IOQ linked to UDT (CDT.NIOQ)
1	multiplexing controller (CDT.MUXC)
2	use standard XIO interface
3	16MB GPMC (CDT.XGPS)
4	cache controller (CDT.CAC)
5	H.F8XIO has determined if the controller is pre-8512-2 or not (CDT.CKFL)
6	controller not pre-8512-2 (CDT.FLOW)
7	reserved for FMS

L.4 Dispatch Queue Entry (DQE)

The dispatch queue entry (DQE) contains all of the core-resident information required to describe an active task to the system. It is always linked to the CPU scheduler state chain that describes the current execution status of the associated task.

Word No. (Decimal)	Byte (Hex)	0 7 8 15 16 23 24 31			
0	0	DQE.SF			
1	4	DQE.SB			
2	8	DQE.CUP	DQE.BUP	DQE.IOP	DQE.US
3	C	DQE.NUM/DQE.TAN			
4-5	10	DQE.ON			
6-7	18	DQE.LMN			
8-9	20	DQE.PSN			
10	28	DQE.USW			
11	2C	DQE.USHF			
12	30	DQE.MSD			
13	34	DQE.KCTR			
14	38	DQE.MMSG	DQE.MRUN	DQE.MNWI	DQE.GQFN
15	3C	DQE.UF2	DQE.IPUF	DQE.NWIO	DQE.SOPO
16	40	DQE.CQC			
17	44	DQE.SH	DQE.SHF	DQE.TIFC	DQE.RILT
18	48	DQE.UTS1			
19	4C	DQE.UTS2			
20	50	DQE.DSW			
21	54	DQE.PRS			
22	58	DQE.PRM			
23	5C	Reserved	DQE.TSKF	DQE.MSPN	DQE.MST
24	60	DQE.PSSF			
25	64	DQE.PSSB			
26	68	DQE.PSPR	DQE.PSCT	DQE.ILN	DQE.RESU
27	6C	DQE.TISF			
28	70	DQE.TISB			
29	74	DQE.TIPR	DQE.TICT	DQE.SWIF	DQE.UBIO
30	78	DQE.RRSF			
31	7C	DQE.RRSB			
32	80	DQE.RRPR	DQE.RRCT	DQE.NSCT	
33	84	DQE.MRSF			
34	88	DQE.MRSB			

Word No. (Decimal)	Byte (Hex)	0 7	8 15	16 23	24 31
35	8C	DQE.MRPR	DQE.MRCT	DQE.NWRR	DQE.NWMR
36	90	DQE.RTI	DQE.NWLM	DQE.ATI	Reserved
37	94	DQE.SAIR/DQE.TAD			
38-40	98	DQE.ABC			
41	A4	DQE.TSAP			
42-43	A8	DQE.SRID/DQE.PGOL			
	AC	DQE.SRID/DQE.PGOC		DQE.SRID/Reserved	
44-51	B0	DQE.CDIR/DQE.CVOL			
52	D0	DQE.GID	Reserved	DQE.ASH	
53	D4	DQE.ACX2			
54	D8	DQE.MRQ	DQE.MEM	DQE.MEMR	
55	DC	DQE.MRT	Reserved	DQE.RMMR	
56	E0	DQE.MAPN		DQE.CME	
57	E4	DQE.CMH		DQE.CMS	
58-63	FC	Reserved			

Byte (Hex)	Symbol	Description
0	DQE.SF	String forward linkage address; Field length = 1W; Standard linked list format; Contains address of next (top-to-bottom) entry in chain.
4	DQE.SB	String backward linkage address; Field length = 1W; Standard linked list format; Contains address of next (bottom-to-top) entry in chain.
8	DQE.CUP	Current user priority; Standard linked list format; This priority is adjusted for priority migration based on situational priority increments. Situational priority increments are based on the base level priority (DQE.BUP) of the task.
	DQE.BUP	Base priority of user task; Field length = 1B; Used by scheduler to generate DQE.CUP (current priority) based on any situational priority increments.
	DQE.IOP	I/O priority; Field length = 1B; Initially set from base priority; Used for I/O queue priority.

Byte (Hex)	Symbol	Description
	DQE.US	State chain index for this user task; Field length = 1B; Range: zero through X'1E'; Indicates current state of this task, such as ready-to-run priority, I/O wait, resource block, etc.

Label	Index	Task description
FREE	00	DQE is available (in free list)
PREA	01	activation in progress
CURR	02	currently executing task or is pre-empted time-distribution task in quantum stage one
SQRT	03	ready to run (priority level 1 to 54)
SQ55	04	ready to run (priority level 55)
SQ56	05	ready to run (priority level 56)
SQ57	06	ready to run (priority level 57)
SQ58	07	ready to run (priority level 58)
SQ59	08	ready to run (priority level 59)
SQ60	09	ready to run (priority level 60)
SQ61	0A	ready to run (priority level 61)
SQ62	0B	ready to run (priority level 62)
SQ63	0C	ready to run (priority level 63)
SQ64	0D	ready to run (priority level 64)
SWTI	0E	waiting for terminal input
SWIO	0F	waiting for I/O
SWSM	10	waiting for message complete
SWSR	11	waiting for run request complete
SWLO	12	waiting for low speed output
SUSP	13	waiting for timer expiration, resume request, or message interrupt
RUNW	14	waiting for timer expiration, or run request
HOLD	15	waiting for a continue request
ANYW	16	waiting for timer expiration, no-wait I/O complete, no-wait message complete, no-wait run request complete, message interrupt, or break interrupt
SWDC	17	waiting for disk space
SWDV	18	waiting for device allocation
SWFI	19	waiting for file system
MRQ	1A	waiting for memory
SWMP	1B	waiting for memory pool
SWGQ	1C	waiting in general wait queue
CIPU	1D	current IPU task in execution
RIPU	1E	IPU requesting state

Byte (Hex)	Symbol	Description
C	DQE.NUM	DQE entry number; Field length = 1B; Used as an index to DQE address table (DAT); Range: one through "N"(for MPL index compatibility); Used by scheduler to set C.PRNO to reflect the currently executing task. This value is also used as the MPL index. It is used by the scheduler to initialize the CPIX in the PSD before loading the map for this task.
	DQE.TAN	Task activation sequence number; Field length = 1W; This number is assigned by the activation service and uniquely identifies a task. **Note:** The most significant byte of this value is the DQE entry number and is accessible as DQE.NUM.
10	DQE.ON	Owner name; Field length = 1D.
18	DQE.LMN	Load module name; Field length = 1D.
20	DQE.PSN	Pseudonym associated with task; Field length = 1D; This parameter is an optional argument accepted by the pseudo task activation service. It can be used to uniquely identify a task within a subsystem, such as multibatch. It contains descriptive information useful to the system operator or to other tasks within a subsystem. Conventions used to generate a pseudonym are determined by the associated subsystem. A system-wide convention should be used to establish pseudonym prefix conventions to avoid confusion between subsystems.
28	DQE.USW	User status word; Field length = 1W.
2C	DQE.USHF	Scheduling flags; Field length = 1W; Used by the scheduler to indicate special status conditions.

Byte (Hex)	Symbol	Description

Bit	Meaning When Set
00	load protection image requested (DQE.LPI)
01	single copy load module (DQE.SING)
02	task is indirectly connected (DQE.INDC)
03	task is privileged (DQE.PRIV)
04	task has message receiver (DQE.MSGR)
05	task has break receiver (DQE.BRKR)
06	task quantum stage one expired (DQE.QS1X)
07	task quantum stage two expired (DQE.QS2X)
08	in-swap I/O error (DQE.INER)
09	wait I/O request outstanding (DQE.WIOA)
10	wait I/O complete before in-progress notification (DQE.WIOC)
11	inhibit message pseudointerrupt (DQE.INMI)
12	batch origin task (DQE.BAOR)
13	running in TSM environment (DQE.TMOR)
14	task abort in progress (DQE.ABRT)
15	task is in pre-exit state (DQE.PRXT)
16	run receiver mode (DQE.RRMD)
17	wait send message outstanding (DQE.WMSA)
18	wait message complete before link to wait queue (DQE.WMSC)
19	wait mode send run request outstanding (DQE.WRRA)
20	wait mode send run request complete before link to wait queue (DQE.WRRC)
21	debug associated with task (DQE.DBAT)
22	real-time task (DQE.RT)
23	time-distribution task initial dispatch (DQE.TDID) Set by: • H.ALOC1 on activation of T/D task. • S.EXEC51 when task is linked to wait state. • H.EXEC7 on completion of inswap or other memory request. Reset by: • S.EXEC20 on initial dispatch of task after activation • Wait state termination • In-swap
24	task delete in progress (DQE.DELP)
25	task abort (with abort receiver) in progress (DQE.ABRA)
26	abort receiver established (DQE.ABRC)
27	asynchronous abort/delete inhibited (DQE.ADIN)
28	asynchronous delete deferred (DQE.ADDF)
29	task is inactive (DQE.INAC)
30	asynchronous abort deferred (DQE.AADF)
31	activation timer in effect (DQE.ACTT)

Byte (Hex)	Symbol	Description
30	DQE.MSD	Physical address of MIDL in TSA; Field length = 1W.
34	DQE.KCTR	Kill/abort timer; Field length = 1W.
38	DQE.MMSG	Maximum number of no wait messages allowed to be sent by this task; Field length = 1B.
	DQE.MRUN	Maximum number of no-wait run requests allowed to be sent by this task; Field length = 1B.
	DQE.MNWI	Maximum number of no-wait I/O requests allowed to be concurrently outstanding for this task; Field length = 1B.
	DQE.GQFN	Contains the generalized queue (SWGQ) function code; Field length = 1B; Function codes are queued as follows:

Code	Meaning
01	volume resource (QVRES)
02	ART space (QART)
03	mount in progress (QMNT)
04	resourcemark lock (QRSM)
05	reserved for eventmark (QEVM)
06	read wait for writer (QGEN)
07	shared memory table (QSMT)
08	synchronous resource lock (QSRL)
09	mounted volume table (QMVT)
0A	dual-port lock (QDPLK)
0B	suspend dual-port lock (QSUSP)
0C	debug wait (QDBGW)
0D	remote message area (QMSG)
0E	remote message event (QSER)
0F	remote allocate area (QASMP)
10	remote deallocate area (QDSMP)
11	remote abort area (QAMSG)
12	remote enable/disable area (QOMSG)
13	wait for TSM (QWTSM)

Byte (Hex)	Symbol	Description
3C	DQE.UF2	Scheduling flags; Field length = 1B;

Bit	Meaning if Set
0	enable debug mode break (DQE.EDB)
1	generalized wait queue time-out (DQE.GQTO)
2	task interrupts are synchronized (DQE.SYNC)
3	task is part of a job (DQE.JOB)
4	ACX-32 task flag (DQE.ACX)
5	special arithmetic function requested (DQE.AF)
6	reserved
7	run request terminated (DQE.RRT)

DQE.IPUF — IPU flag byte;
Field length = 1B;

Bit	Meaning if Set
0	IPU inhibit flag (DQE.IPUH)
1	IPU bias flag (DQE.IPUB)
2	CPU only (DQE.IPUR)
3	OS execution direction flag (set when PSD is in user area) (DQE.OSD)
4	base register task (DQE.BASE)
5	Ada task (DQE.ADA)
6	PTRACE debugger task (DQE.PDBG)
7	H.PTRAC task association control bit (DQE.PTRA)

DQE.NWIO — Number of no-wait I/O requests;
Field length = 1B.

DQE.SOPO — Priority bias only swapping control flags;
Field length = 1B;

Bit	Meaning if Set
0	SWGQ state priority-based swapping (DQE.GQPO)
1	swap inhibit due to bit map access (DQE.BMAP)
2	inhibit swap device while accessing MDT (DQE.MDTA)
3	user swap inhibit flag (DQE.USWI)
4	user swap on priority only flag (DQE.USPO)
5-7	reserved

Byte (Hex)	Symbol	Description
40	DQE.CQC	Current quantum count; Field length = 1W; Used by the scheduler to accumulate elapsed execution time for the task to compare the level unique stage one and stage two time-distribution values.
44	DQE.SH	Used by J.SWAPR to swap shadow memory; Field length = 1B.
	DQE.SHF	Shadow memory flag; Field length = 1B; Bit — Meaning if Set 0 — task requests shadow memory (DQE.SHAD) 1 — IPU shadow memory requested (DQE.SHI) 2 — IPU/CPU Common Shadow Memory requested (DQE.SHB).
	DQE.TIFC	Timer function code; Field length = 1B; Value — Meaning 00 — not active 01 — request interrupt 02 — resume program from suspend (SUSP) queue 03 — resume program from any-wait (ANYW) queue 04 — resume program from run-request-wait (RUNW) queue 05 — resume program from generalized (SWGQ) queue 06 — resume program from peripheral device (SWDV) queue 07 — resume program from disk space (SWDC) queue
	DQE.RILT	Request Interrupt (RI) level for timer; Field length = 1B; Identifies the interrupt level to be requested upon timer expiration.
48	DQE.UTS1	User timer slot word 1; Field length = 1W; Current timer value; Contains negative number of timer units before time out.

Byte (Hex)	Symbol	Description
4C	DQE.UTS2	User timer slot word 2; Field length = 1W; Reset timer value; Contains negative number of timer units; Used to reset the current timer value when it expires.
50	DQE.DSW	Base mode debugger status word (PCALL); Field length = 1W.
54	DQE.PRS	Peripheral requirement specification; Field length = 1W;

Bit	Description
0-7	reserved
8-15	device type code
16-23	channel address
24-31	subchannel address or contains first word of SWGQ ID.

Byte (Hex)	Symbol	Description
58	DQE.PRM	Peripheral requirements mask; Field length = 1W;

Value	Meaning
X'00FF0000'	any device of this type code
X'00FFFF00'	any device of the specified type code on the specified channel
X'00FFFFFF'	the specified device as described by type code, channel, and subchannel address, or contains second word of SWGQ ID.

Byte (Hex)	Symbol	Description
5C	Reserved	Field length = 1B
	DQE.TSKF	Task flags; Field length = 1B;

Bit	Meaning if Set
0	real-time accounting disabled (DQE.RTAC)
1-2	reserved for RMSS
3	task is running with MPX-32 mapped out (DQE.MAPO)
4	reserved for MPX-32
5	task is demand paged (DQE.DPG)
6	inhibit page out (DQE.NPGO)
7	reserved

Byte (Hex)	Symbol	Description
	DQE.MSPN	TSA maps required to span MIDLs and MEMLs; Field length = 1B.
	DQE.MST	Static memory type specification; Field length = 1B; Value — Memory Class 01 — E 02 — H 03 — S 04 — H1 05 — H2 06 — H3 This field is used to specify the type of memory required for in-swap.
60	DQE.PSSF	Pre-emptive system service head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
64	DQE.PSSB	Pre-emptive system service head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
68	DQE.PSPR	Pre-emptive system service head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.PSCT	Pre-emptive system service head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.ILN	Interrupt level number; Field length = 1B; Identifies associated interrupt level for interrupt connected tasks.
	DQE.RESU	Reserved usage index; Field length = 1B.

Byte (Hex)	Symbol	Description
6C	DQE.TISF	Task interrupt head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
70	DQE.TISB	Task interrupt head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
74	DQE.TIPR	Task interrupt head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.TICT	Task interrupt head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.SWIF	Swapping inhibit flags; Field length = 1B;

Bit	Task Meaning if Set
0	resident (DQE.RESP)
1	locked in memory (DQE.LKIM)
2	unbuffered I/O in progress (DQE.IO)
3	outswapped (DQE.OTSW)
4	leaving system (DQE.TLVS)
5	forced unswappable during terminal output (DQE.FCUS)
6	forced unswappable because swap file has not been allocated for it (DQE.FCRS)
7	imbedded in the operating system (DQE.INOS)

	DQE.UBIO	Number of unbuffered I/O requests currently outstanding; Field length = 1B.
78	DQE.RRSF	Run receiver head cell string forward linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
7C	DQE.RRSB	Run receiver head cell string backward linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
80	DQE.RRPR	Run receiver head cell dummy priority (always zero); Standard head cell format; Field length = 1B.

Byte (Hex)	Symbol	Description
	DQE.RRCT	Run receiver head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.NSCT	Number of map blocks outswapped; Field length = 1H.
84	DQE.MRSF	Message receiver head cell string forward Linkage address; Standard head cell format; Field length = 1W; Contains address of next (top-to-bottom) entry in chain.
88	DQE.MRSB	Message receiver head cell string backward Linkage address; Standard head cell format; Field length = 1W; Contains address of next (bottom-to-top) entry in chain.
8C	DQE.MRPR	Message receiver head cell dummy priority (always zero); Standard head cell format; Field length = 1B.
	DQE.MRCT	Message receiver head cell number of entries in list; Standard head cell format; Field length = 1B.
	DQE.NWRR	Number of no-wait mode run requests outstanding; Field length = 1B.
	DQE.NWMR	Number of no-wait mode message requests outstanding; Field length = 1B.
90	DQE.RTI	Requested task interrupt flags; Field length = 1B;

Bit	Meaning if Set
0	reserved
1	priority one end action request. Used for pre-emptive system services. (DQE.EA1R)
2	debug break request (DQE.DBBR)
3	user break request (DQE.UBKR)
4	priority two end action request (DQE.EA2R)
5	message interrupt request (DQE.MSIR)
6-7	reserved

Byte (Hex)	Symbol	Description
	DQE.NWLM	No-wait run request limit. Field length = 1B.
	DQE.ATI	Active task interrupt flags; Field length = 1B; Bit — Meaning if Set 0 — reserved 1 — priority one active end action (DQE.AEA1) 2 — active debug break (DQE.ADM) 3 — active user break (DQE.AUB) 4 — priority two active end action (DQE.AEA) 5 — active message interrupt (DQE.AMI) 6-7 — reserved
	Reserved	Field length = 1B.
94	DQE.SAIR	System action task interrupt request; Bit — Meaning if Set 0 — request for delete of this task (DQE.DELR) 1 — reserved 2 — hold task request (DQE.HLDR) 3 — abort task request (DQE.ABTR) 4 — exit task request (DQE.EXTR) 5 — suspend task request (DQE.SUSR) 6 — run receiver mode request (DQE.RRRQ) 7 — reserved
	DQE.TAD	TSA address (logical); Field length = 1W; Byte zero contains DQE.SAIR.
98	DQE.ABC	Abort code; Field length = 3W.
A4	DQE.TSAP	Physical address of the TSA
A8-AC	DQE.SRID	If DQE.DPG is reset; Used swap space linked list; Field length = 2W.
	DQE.PGOL	If DQE.DPG is set; Forward pointer to MPTL (MAP.SF); Field length = 1HW Backward pointer to MPTL (MAP.SB) Field length = 1HW
	DQE.PGOC	Number of pages queued for pageout Field length = 1HW
	Reserved	Field length = 1HW

Byte (Hex)	Symbol	Description
B0	DQE.CDIR	Load module RID at activation; Field length = 8W.
	DQE.CVOL	Current working volume at activation; Field length = 8W.
D0	DQE.GID	Group swap identification; Field length = 1B.
D1	Reserved	1 Byte
D2	DQE.ASH	Number of shadow memory blocks currently allocated Field length = 1H.
D4	DQE.ACX2	Advance communication word; Field length = 1W.
D8	DQE.MRQ	Memory request doubleword; Reserved field length = 1B.
	DQE.MEM	Type of memory requested; Field length = 1B;

Value	Memory Class
01	E
02	H
03	S

	DQE.MEMR	Number of memory blocks required; Field length = 1H.
DC	DQE.MRT	Memory request type code; Field length = 1B;

Value	Meaning
00	in-swap only
01	preactivation request
02	activation request
03	memory expansion request
04	IOCS buffer request
05	shared memory request
06	system buffer request
07	release swap file space

If DQE.MRT equals 05, the next three bytes will contain the address of the shared memory table entry.

	Reserved	Field length = 1B.
	DQE.RMMR	Map register for requested memory; Field length = 1H.

Byte (Hex)	Symbol	Description
E0	DQE.MAPN	Inclusive span of maps in use; Field length = 1H.
	DQE.CME	Number of swappable class E map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
E4	DQE.CMH	Number of swappable class H map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
	DQE.CMS	Number of swappable class S map blocks currently allocated; For resident tasks, if not zero, reflects the total number of map blocks in use. Field length = 1H.
E8	Reserved	Reserved for MPX-32

L.5 File Control Block (FCB), 16 Word

Word	0 – 7	8 – 12	13 – 31
0	**Opcode (FCB.OPCD)**	Logical file code (FCB.LFC)	
1	**Reserved**		
2	General control flags (FCB.GCFG)	Special flags (FCB.SCFG)	**Reserved**
3	**Status flags (FCB.SFLG)**		
4	**Actual transfer quantity (FCB.RECL)**		
5	**Reserved**	**I/O queue address (FCB.IOQA)**	
6	**Special Status (FCB.SPST)**	Wait I/O error return address (FCB.ERRT)	
7	**Index to FPT (FCB.FPTI)**	**FAT address (FCB.FATA)**	
8	**Reserved**	I/O buffer address (FCB.ERWA)	
9	Transfer quantity (bytes) (FCB.EQTY)		
10	Random access address (FCB.ERAA)		
11	**Extended I/O status word one (FCB.IST1)**		
12	**Extended I/O status word two (FCB.IST2)**		
13	**Reserved**	No-wait I/O normal end-action service address (FCB.NWOK)	
14	**Reserved**	No-wait I/O error end-action service address (FCB.NWER)	
15	Number of buffers (FCB.BBN)	Address of blocking buffer (FCB.BBA)	

Shaded areas are set by the system.

T1FCB

Word 0

Bit 0 Reserved

Bits 1-7 Operation code (FCB.OPCD) — type of function requested of the device handler. This field is set by IOCS as a function of the executed service.

Bits 8-31 Logical file code (FCB.LFC) — any combination of three ASCII characters is allowed. The LFC must match the previously assigned LFC of the I/O resource being accessed.

Word 1

Bits 0-31 Reserved

Word 2

Bits 0-7 General control flags (FCB.GCFG) — these eight bits enable the user to specify the manner in which an operation is to be performed by IOCS. The interpretation of these bits is shown as follows:

Bit	Meaning if Set	Definition
0	NWT	IOCS returns to the user immediately after the I/O operation is queued. If reset, IOCS exits to the calling program only when the requested operation has been completed.
1	NER	error processing is not performed by either the device handler or IOCS. An error return address is ignored and a normal return is taken to the caller; however, the device status is posted in the FCB unless bit 3 is set. If reset, normal error recovery is attempted. Normal error processing for disk and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message INOP to the console, which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disk or magnetic tape is unsuccessful, an error status message is typed to the console and the error return address is taken if provided. Otherwise, the task is aborted.
2	DFI	data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. See Table L-1 for more explanation.
3	NST	device handlers perform no status checking and no status information is returned. All I/O appears to complete without error. Otherwise, status checking is performed and status information is returned as necessary.
4	RAN	file accessing occurs in the random mode. Otherwise, sequential accessing is performed. Note: This bit is set if word 2 bit 12 is set.
5		reserved (M.FILE)
6	EXP	must be 1 for 16-word FCB.
7	IEC	this bit is reserved for internal IOCS use.

Bits 8-12 Special Control Specification (FCB.SCFG). — This field contains device control specifications unique to certain devices. Interpretation and processing of these specifications are performed by the device handlers. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table L-1.

Bits 13-31 reserved for extended control specifications

Bit	Meaning if Set	Definition
13	RXON	software read flow control required for 8-Line ACM (FCB.RXON)

Table L-1
Special Control Flags

Device	Bit 2=0	Bit 2=1	Bit 8=0	Bit 8=1	Bit 9=0	Bit 9=1
Line Printer (LP)	Interpret first character as carriage control	Interpret first character as data See bit 8	Form control	No form control		
Discs, (DM,DF, FL)	Report EOF if X'0FE0FE0F' encountered in word 0 of 1st block during read of unblocked file	X'0FE0FE0F' in word 0 not recognized as EOF				
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	Perform special character formatting	No special character formatting	ASCII control passed as data	ASCII control character detect	Echo by controller	No echo by controller
	M.WRIT	**M.WRIT**	**SVC 1,X'3E'**	**SVC 1,X'3E'**	**M.WRIT**	**M.WRIT**
	Interpret first character as carriage control	Interpret first character as data	Stop transmitting break	Start transmitting break	Normal write	Initialize device (load UART parameters)

Device	Bit 10=0	Bit 10=1	Bit 11=0	Bit 11=1	Bit 12=0	Bit 12=1
Line Printer (LP)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Discs, (DM,DF, FL)					Normal read	Read with byte granularity (word 2 bit 4 set)
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	(If bit 2=0) convert lower case character to upper case	Inhibit conversion	No special character detect	Special character detect	Do not purge type ahead buffer	Purge type ahead buffer
	M.WRIT	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**
			Normal write	Write with input sub-channel monitoring plus software flow control		

Continued on next page

Table L-1
Special Control Flags (*Continued*)

Device	(Bit 2=0)	(Bit 2=1)	Bit 8	Bit 9		Bit 10	Bit 11	Bit 12
ALIM (Asynch-ronous Line Interface Module) Terminals (TY)	Read: receive data (bytes) defined for transfer count	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
		0	1	0	=Blind mode reset	1= Inhibit conversion of lower case characters to upper case		
		0	0	1	=Echo on read			
		1	N/A	N/A	=Receive data			
		0	0	0	=Receive data			
					Write			
	Write: formatted	0	N/A	0	=Formatted write	0= Convert		
		0	N/A	1	=Initialize device			
		1	N/A	N/A	=Unformatted write			

Word 3

Bits 0-31 Status word (FCB.SFLG) — 32 indicator bits are set by IOCS to indicate the status, error, and abnormal conditions detected during the current or previous operation. The assignment of these bits is shown as follows:

Bits	Meaning if Set	Definition
0	OP	operation in progress. Request has been queued. (Note: Reset after post I/O processing complete.)
1	ERR	error condition found
2	BB	invalid blocking buffer control pointers have been encountered during file blocking or unblocking
3	PRO	write protect violation
4	INOP	device inoperable
5	BOM	beginning-of-medium (BOM) (load point) or illegal volume number (multivolume magnetic tape)
6	EOF	end-of-file
7	EOM	end-of-medium (end of tape, end of disk file)
8-9		reserved
10	TIME	last command exceeded time-out value and was terminated
11-15		reserved
16	ECHO	echo
17	INT	post program-controlled interrupt
18	LEN	incorrect length
19	PROG	channel program check
20	DATA	channel data check
21	CTRL	channel control check
22	INTF	interface check
23	CHAI	chaining check
24	BUSY	busy
25	ST	status modified
26	CTR	controller end
27	ATTN	attention
28	CHA	channel end
29	DEV	device end
30	CHK	unit check
31	EXC	unit exception

Word 4

Bits 0-31 Record length (FCB.RECL) — this field is set by IOCS to indicate the actual number of bytes transferred during read/write operations.

Word 5

Bits 0-7 Reserved

Bits 8-31 I/O queue address (FCB.IOQA) — this field is used by IOCS to point to the I/O queue for an I/O request initiated from this FCB

Word 6

Bits 0-7 Special status bits (FCB.SPST). The interpretation of these bits is shown below:

Bits	Definition
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	request killed, I/O not issued
3	if set, exceptional condition has occurred in the I/O request
4	if set, software read flow control required
5-7	reserved

Bits 8-31 Wait I/O error return address (FCB.ERRT) — this field is set by the user and contains the address to which control is to be transferred in the case of an unrecoverable error when control bits 1 and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the requesting task is aborted.

Word 7

Bits 0-7 Index to FPT (FCB.FPTI) — this field is set by IOCS to index into the associated entry in the file pointer table (FPT)

Bits 8-31 FAT address (FCB.FATA) — this field is set by IOCS to point to the associated file assignment table (FAT) entry.

Word 8

Bits 0-7 Reserved

Bits 8-31 Data buffer address (FCB.ERWA) — start address of data area for read or write operations. (24 bit pure address)

Word 9

Bits 0-31 Quantity (FCB.EQTY)— number of bytes of data to be transferred

Word 10

Bits 0-31 Random access address (FCB.ERAA) — this field contains a block number (zero origin) relative to the beginning of the disk file. It is the start address for the current read or write operation with word 2 bit 4 set and word 2 bit 12 reset.

or

For disk read requests with word 2 bits 4 and 12 set (read with byte granularity), this word defines the byte offset relative to the beginning of the file. Note: If word 9 is zero, the file retains its position prior to the call.

Word 11

Bits 0-31 Status word one (FCB.IST1) — these are the first 32 bits of status returned by the sense command

Word 12

Bits 0-31 Status word two (FCB.IST2) — these are the second 32 bits of status returned by the sense command

Word 13

Bits 0-7 Reserved

Bits 8-31 No-wait I/O (FCB.NWOK) — normal completion return address. This user routine must be exited by calling the M.XIEA service.

Word 14

Bits 0-7 Reserved

Bits 8-31 No-wait I/O (FCB.NWER) — error completion return address. This user routine must be exited by calling the M.XIEA service.

Word 15 (Applicable only to volume resource.)

Bits 0-7 (FCB.BBN) — Number of 192 word buffers for user supplied blocking buffers. A value of one or zero in this field specifies one blocking buffer.

Bits 8-31 Blocking buffer address (FCB.BBA) — starting address of a contiguous area of memory FCB.BBN buffers long

Table L-2
Device Functions (Standard Devices)

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Open (M.FILE)	0	IOCS opens	IOCS opens	IOCS opens	Initialize IOP channel if necessary
Rewind (M.RWND)	1	Eject,set BOM bit word 3 bit 5 in FCB	Rewind Tape	Set current block address to zero (FAT)	SENSE operation
Read Record (M.READ)	2	Spec error	Read to data buffer	Read to data buffer	Read to data buffer
Write record (M.WRIT)	3	Write from data buffer	Write from data buffer. If blocked writes *n* data buffers to blocking buffer before output	Write from data buffer. If blocked IOCS writes *n* data buffers to blocking buffer before output	Write record to terminal
Write EOF (M.WEOF)	4	NOP*	Write EOF	If blocked, IOCS writes EOF. If unblocked writes X'0FE0FE0F'	NOP*
Execute Channel	5	Spec error	Execute Channel Program	Execute Channel Program	Execute channel Program

*NOP — No operation performed

Continued on next page

Table L-2
Device Functions (Standard Devices) *(Continued)*

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Advance Record (M.FWRD)	6	Spec error	Advance record	If blocked, advance record. If unblocked, advance one 192W block.	Set data terminal ready
Advance File (M.FWRD)	7	Spec error	Advance file (past EOF)	Spec error	Reset data terminal ready
Backspace Record (M.BACK)	8	Spec error	Backspace record	If blocked, backspace record. If unblocked backspace one 192W block	Used by J.TINIT to initialize terminals
Backspace File (M.BACK)	9	Spec error	Backspace file to previous EOF	Spec error	Reset request to send command
Upspace (M.UPSP)	A	Upspace	Multivolume only. If BOT, writes volume record. If EOT, performs ERASE, writes EOF, and issues MOUNT message.	Spec error on F-class disks. For floppy only: format diskette. New diskettes must be formatted prior to normal usage.	Set request to send command
Erase or Punch Trailer Not user IOCS/handler provides call automatically	B	NOP	Multivolume only. Same as upspace above. Erases 4" of tape before writing	NOP	Set/reset break (depends on flags in FCB)

Continued on next page

Table L-2
Device Functions (Standard Devices) *(Continued)*

Operation	IOCS Op Code	Line Printer (LP)	Mag Tape (M9/MT)	Disk (DM/DF/ DC/Floppy)	Handler=F8XIO (8-Line)
Eject/ Punch Leader (M.EJECT)	C	Eject to top of form	Write dummy record with eject control character as first character	NOP	Define special character
Close (M.CLSE)	D	IOCS closes	IOCS closes	IOCS closes	NOP
Reserve FHD Port	E	Spec error	Spec error	Reserve port-4MB disk only. Else, spec error Reserve Dual Ported Disk	Set single-channel operation (default) command
Release FHD Port	F	Spec error	Spec error	Release port-4MB disk only. Else, spec error Reserve Dual Ported Disk	Set dual-channel operation

Table L-3
Device Functions (Terminals, Handler Action Only)

Operation	IOCS Op Code	Handler = H.ASMP (ALIM)
Open M.FILE	0	NOP*
Rewind M.RWND	1	NOP*
Read record M.READ	2	Read to data buffer
Write record M.WRIT	3	Write record to terminal
Write EOF M.WEOF	4	NOP*
Execute channel	5	Execute channel
Advance record M.FWRD	6	Connect communications channel
Advance file M.FWRD	7	Disconnect communications channel
Backspace record M.BACK	8	Initialize device and set time-out value
Backspace file M.BACK	9	Clear break status flag word
Upspace M.UPSP	A	Spec error**
Erase/punch trailer	B	Transmit break
Eject/punch leader M.EJECT	C	Spec error**
Close M.CLSE	D	NOP*
Reserve FHD port	E	Spec error**
Release FHD port	F	Spec error**

* NOP = No operation performed

** Spec Error = Illegal operation code

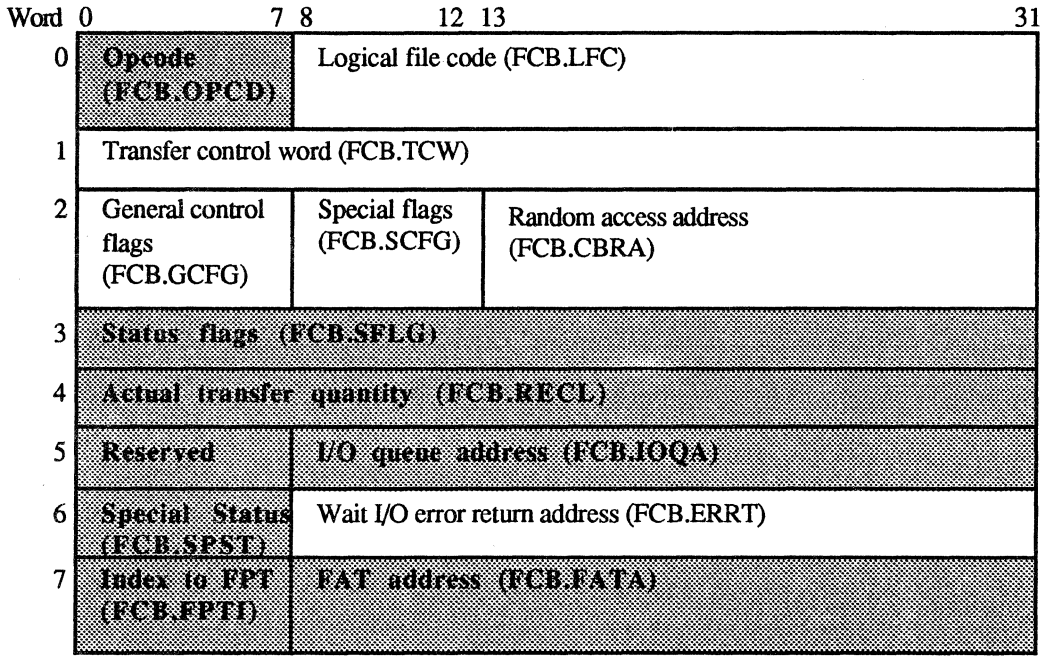
Table L-4
Standard Carriage Control Characters and Interpretation

Control Character	Hexa-decimal Value	Result on a Terminal	Result on Directly Allocated Printer (Serial or parallel)	SLO
Blank	20	One linefeed, one carriage return before write	Single space before print	Single space before print
0	30	Two linefeeds, one carriage return before write	Double space before print	Double space before
1	31	Five linefeeds, one carriage return before write	Page eject (slew) before print	Page eject (slew) before print
+	2B	No linefeed, no carriage return before write (line append)	No space before print (overprint)	No space before print (overprint)
-	2D	Five linefeeds, one carriage return before write	Single space before print	Page eject, save and print up to three user supplied title lines. See Note 1.
<	3C	One linefeed, one carriage return before write	Single space before print	Set inhibit spooler title line in this file.
>	3E	One linefeed, one carriage return before write	Single space before print	Set enable spooler title line in this file.
=	3D	One linefeed, one carriage return before write	Single space before print	Page eject and clear up to three user-supplied title lines in this file.

Notes:

1. User-supplied title lines have the same effect as this character. Supplying a fourth title line clears the first three, but only one page is ejected. User-supplied titles are retained by the spooler and are repeated at the top of each page until cleared or the spool file ends.

L.6 File Control Block, Compatible Mode 8 word (FCB)



Shaded areas are set by the system.

A.L8W.FCB

Word 0

Bit 0 Reserved

Bits 1-7 Operation code (FCB.OPCD) — type of function requested of the device handler. This field is set by IOCS as a function of the requested service.

Bits 8-31 Logical file code (FCB.LFC) — any combination of three ASCII characters is allowed.

Word 1 (FCB.TCW)

This word supplies a transfer control word (TCW) that describes a data buffer and transfer quantity. If no TCW definition is supplied, the transfer buffer defaults to location zero of the task's logical address space and is 4096 words long.

Bits 0-11 Quantity — 12 bit field specifying the number of data items to be transferred. This quantity must include the carriage control character, if applicable. The transfer quantity is in units determined by the address in bits 12 to 31.

Bits 12-31 Format code and buffer address— bits 12, 30 and 31 specify byte, halfword, or word quantities for data transfers. They are interpreted as follows:

Type of Transfer	F (12)	C (30,31)	Address
Byte	1	xx	13-31
Halfword	0	x1	13-30
Word	0	00	13-29

Word 2

Bits 0-7 General control flags (FCB.GCFG) — these eight bits enable the user to specify the manner in which an operation is to be performed by IOCS. The interpretation of these bits is shown below:

Bit	Meaning if Set	Definition
0	NWT	IOCS returns to the user immediately after the I/O operation is queued. If reset, IOCS exits to the calling program only when the requested operation has been completed.
1	NER	error processing is not performed by either the device handler or IOCS. An error return address is ignored and a normal return is taken to the caller; however, the device status is posted in the FCB unless bit 3 is set. If reset, normal error recovery is attempted. Normal error processing for disk and magnetic tape is automatic error retry. Error processing for unit record devices except the system console is accomplished by IOCS typing the message INOP to the console, which allows the operator to retry or abort the I/O operation. If the operator aborts the I/O operation, or if automatic error retry for disk or magnetic tape is unsuccessful, an error status message is typed to the console and the error return address is taken if provided. Otherwise, the task is aborted.
2	DFI	data formatting is inhibited. Otherwise, data formatting is performed by the appropriate device handler. See Table L-5 for more explanation.
3	NST	device handlers perform no status checking and no status information is returned. All I/O appears to complete without error. Otherwise, status checking is performed and status information is returned as necessary.
4	RAN	file accessing occurs in the random mode. Otherwise, sequential accessing is performed.
5		reserved (M.FILE)
6	EXP	must be 0 for 8 word FCB.
7	IEC	this bit is reserved for internal IOCS use.

Bits 8-12 Special Control Specification (FCB.SCFG). — This field contains device control specifications unique to certain devices. Interpretation and processing of these specifications are performed by the device handlers. A bit setting is meaningful only when a particular type of device is assigned as indicated in Table L-2.

Bits 13-31 Random access address (FCB.CBRA) — This field contains a block number (zero origin) relative to the beginning of the disk file, and specifies the base address for read or write operations.

Table L-5
Special Control Flags (8 Word FCB)

Device	Bit 2=0	Bit 2=1	Bit 8=0	Bit 8=1	Bit 9=0	Bit 9=1
Line Printer (LP)	Interpret first character as carriage control	Interpret first character as data See bit 8	Form control	No form control		
Discs, (DM,DF, FL)	Report EOF if X'0FE0FE0F' encountered in word 0 of 1st block during read of unblocked file	X'0FE0FE0F' in word 0 not recognized as EOF				
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	Perform special character formatting	No special character formatting	ASCII control passed as data	ASCII control character detect	Echo by controller	No echo by controller
	M.WRIT	**M.WRIT**	**SVC 1,X'3E'**	**SVC 1,X'3E'**	**M.WRIT**	**M.WRIT**
	Interpret first character as carriage control	Interpret first character as data	Stop transmitting break	Start transmitting break	Normal write	Initialize device (load UART parameters)

Device	Bit 10=0	Bit 10=1	Bit 11=0	Bit 11=1	Bit 12=0	Bit 12=1
Line Printer (LP)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
Discs, (DM,DF, FL)						
8-Line Asynchronous Communications Multiplexer (TY)	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**	**M.READ**
	(If bit 2=0) convert lower case character to upper case	Inhibit conversion	No special character detect	Special character detect	Do not purge type ahead buffer	Purge type ahead buffer
	M.WRIT	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**	**M.WRIT**
			Normal write	Write with input sub-channel monitoring plus software flow control		

Continued on next page

Table L-5
Special Control Flags (8 Word FCB) (*Continued*)

Device	(Bit 2=0)	(Bit 2=1)	Bit 8	Bit 9		Bit 10	Bit 11	Bit 12
ALIM (Asynch-	Read: receive	**Bit 2**	**Bit 8**	**Bit 9**	**Read**	On Read:		
ronous	data	0	1	0	=Blind mode reset	1= Inhibit		
Line	(bytes)	0	0	1	=Echo on read	conversion		
Interface	defined	1	N/A	N/A	=Receive data	of lower		
Module)	for	0	0	0	=Receive data	case		
Terminals	transfer					characters		
(TY)	count					to upper		
					Write	case		
	Write:	0	N/A	0	=Formatted write	0= Convert		
	formatted	0	N/A	1	=Initialize device			
		1	N/A	N/A	=Unformatted write			

Word 3

Bits 0-31 Status word (FCB.SFLG) — 32 indicator bits are set by IOCS to indicate the status, error, and abnormal conditions detected during the current or previous operation. The assignment of these bits is shown as follows:

Bits	Meaning if Set	Definition
0	OP	operation in progress. Request has been queued. (Note: Reset after post I/O processing complete.)
1	ERR	error condition found
2	BB	invalid blocking buffer control pointers have been encountered during file blocking or unblocking
3	PRO	write protect violation
4	INOP	device inoperable
5	BOM	beginning-of-medium (BOM) (load point) or illegal volume number (multivolume magnetic tape)
6	EOF	end-of-file
7	EOM	end-of-medium (end of tape, end of disk file)
8-9		reserved
10	TIME	last command exceeded time-out value and was terminated
11-15		reserved
16	ECHO	echo
17	INT	post program-controlled interrupt
18	LEN	incorrect length
19	PROG	channel program check
20	DATA	channel data check
21	CTRL	channel control check
22	INTF	interface check
23	CHAI	chaining check
24	BUSY	busy
25	ST	status modified
26	CTR	controller end
27	ATTN	attention
28	CHA	channel end
29	DEV	device end
30	CHK	unit check
31	EXC	unit exception

Word 4

Bits 0-31 Record length (FCB.RECL) — this field is set by IOCS to indicate the actual number of bytes transferred during read/write operations.

Word 5

Bits 0-7 Reserved

Bits 8-31 I/O queue address (FCB.IOQA) — this field is used by IOCS to point to the I/O queue for an I/O request initiated from this FCB

Word 6

Bits 0-7 Special status bits (FCB.SPST). The interpretation of these bits is shown below:

Bits	Definition
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	kill command, I/O not issued
3	if set, exceptional condition has occurred in the I/O request
4	if set, software read flow control required
5-7	reserved

Bits 8-31 Wait I/O error return address (FCB.ERRT) — this field is set by the user and contains the address to which control is to be transferred in the case of an unrecoverable error when control bits 1 and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the user is aborted.

Word 7

Bits 0-7 Index to FPT (FCB.FPTI) — this field indexes into the appropriate entry in the file pointer table (FPT)

Bits 8-31 FAT address (FCB.FATA) — this field points to the file assignment table (FAT) entry associated with all I/O performed for this FCB. This field is supplied by IOCS.

L.7 File Control Block (FCB), High Speed Data

The following section details the 16 words that make up the FCB for the HSD.

Word	0 7	8 15	16 23	24 31
0	Opcode (FCB.OPCD)	Logical file code (FCB.LFC)		
1	Reserved			
2	General control flags (FCB.GCFG)	Special flags (FCB.SCFG)	Reserved	UDDCMD of IOCD if bit 11 of word 2 is set
3	Status flags (FCB.SFLG)			
4	Record Length in bytes (FCB.RECL)			
5	Reserved	I/O queue address (FCB.IOQA)		
6	Special Status (FCB.SPST)	Wait I/O error return address (FCB.ERRT)		
7	Index to FPT (FCB.FPTI)	FAT address (FCB.FATA)		
8	Reserved	Data address (FCB.ERWA)		
9	Transfer quantity (bytes) (FCB.EQTY)			
10	Device command for non-EXCPM (FCB.ERAA)			
11	Reserved			
12	Extended I/O status word two (FCB.IST2)			
13	Reserved	No-wait I/O normal end-action service address (FCB.NWOK)		
14	Reserved	No-wait I/O error end-action service address (FCB.NWER)		
15	Reserved			

Shaded areas are set by the system.

T2FCB

Word 0

Bit 0 Reserved

Bits 1-7 Contain the operation code, set by IOCS that specifies the type of function requested of H.HSDG.

Bits 8-31 Contain the logical file code associated with the device for the I/O operation.

Word 1

This word is reserved and should be set to zero.

Word 2

Bits 0-7 Contain control flags that enable the user to specify how an operation is to be performed by IOCS. Following is the meaning of these bits when they are set:

Bit	Meaning When Set
0	IOCS returns to the user immediately after the I/O operation is queued (no wait I/O). If reset, IOCS exits to the calling program only when the HSD completes the requested operation (wait I/O).
1	H.HSDG and IOCS do not perform error processing. IOCS ignores the error return address and takes a normal return to the caller. H.HSDG posts device status in the FCB (unless bit 3 is set). If reset, H.HSDG and IOCS perform error processing.
2	specifies physical execute channel program. If reset, specifies logical channel program or non-execute channel program I/O request.
3	IOCS performs no status checking and does not return status information. All I/O appears to complete without error. If reset, IOCS performs status checking and returns status information.
4, 5	Reserved, should be zero.
6	specifies 16 word FCB. Must be set to 1.
7	reserved for internal IOCS use.

Bits 8-23 contain the following special flags:

Bit	Meaning When Set
8	specifies request device status after a transfer. H.HSDG adds an IOCB to the IOCL to retrieve device-specific status after the data transfer completes.
9	specifies send device command prior to data transfer. H.HSDG prefixes the transfer with an IOCB that sends a device command word to the device. The value sent is the 32-bits contained in word 10 of the FCB.
10	specifies disable time out for this request. This bit indicates the operation will take an indeterminable period of time. In most cases this applies only to read operations.
11	specifies set UDDCMD from the least significant byte of word 2. This bit indicates that the UDDCMD byte in the data transfer IOCB must be set to the least significant byte of FCB word 2. This allows the user to pass additional control information to the device without modifying the device driver.
12	specifies disable asynchronous status notification during no-wait I/O.
13	specifies the execute channel program request INIT. By setting this bit, all preliminary I/O data structures are set up based on the I/O command list address provided in word 8 of the FCB. When set, this bit prepares for future cyclic I/O requests but does not issue any I/O.
14	specifies the execute channel program request GO. This bit issues an SIO for the most recently processed INIT execute channel program request (see bit 13).
15-23	reserved
Note:	For further information on the HSD FCB please see the H.HSDG section in the MPX-32 Technical Manual Volume II.

Bits 24-31 if bit 11 is set, these bits define the UDDCMD field of the generated IOCB, overriding the default value from a handler table. This field applies only to FCB format.

Word 3

IOCS uses this word to indicate status, error, and abnormal conditions detected during the current or previous operation. Following is the meaning of the bits when they are set:

Bit	Meaning When Set
0	operation in progress. Request has been queued. This bit is reset after post I/O processing completes.
1	error condition found
2, 3	not applicable, should never be returned
4	device inoperable, HSD not present or offline
5-15	not applicable, should never be returned
16	a time-out occurred and a CD terminate was issued.
17, 18	not applicable, should never be returned
19	there was data remaining in the HSD fifo when the transfer count equaled zero.
20	a parity error occurred during the current data transfer.
21	a non-present memory error occurred during the current data transfer.
22	program violation. An invalid operation code was detected.
23	device inoperative
24	HSD data buffer overflow. Some data from the device was lost.
25	external termination
26	IOCB address error
27	error on TI address fetch
28	device EOB
29	Non-device access errors precluded request queuing. For a list of the errors, see word 12.
30, 31	non-execute channel program type of IOCB in error as follows:

Value	Meaning
00	data transfer
01	device status
10	command transfer

Word 4

This word specifies the record length. For non-execute channel program I/O, IOCS sets this word to indicate the number of bytes transferred during read or write operations.

Word 5

Bits 0-7 reserved

Bits 8-31 specify the IOQ address. IOCS sets this field to point to the IOQ entry initiated from this FCB.

Word 6

Bits 0-7 specify special status as follows:

Bit	Meaning When Set
0	no-wait normal end action not taken
1	no-wait error end action not taken
2	kill command, I/O not issued
3	an exception condition has occurred in the I/O request
4	not used
5-7	reserved

Bits 8-31 contain the wait I/O error return address. The user sets this field to the address where control is to be transferred for unrecoverable errors when bits 0, 1, and 3 of word 2 are reset. If this field is not initialized and an unrecoverable error is detected under the above conditions, the user task is aborted.

Word 7

Bits 0-7 set by the I/O control system (IOCS), contains an index to the file pointer table (FPT) entry for this I/O operation.

Bits 8-15 supplied by the IOCS, points to the file assignment table (FAT) entry associated with this FCB.

Word 8

Bits 0-7 reserved

Bits 8-31 these bits are used as the data address, a logical IOCL address, or a physical IOCL address as follows:

Data address – This is the starting address of the data area for FCB format I/O operations. This address must be a word address.

Logical IOCL address – This is a logical, doubleword address that points to a user-supplied IOCL for SIO format I/O operations. For more information about SIO format, refer to Reference Manual Volume I, Chapter 3. The execute channel program entry point (H.IOCS,10) must be used and bit 2 of word 2 of the FCB is reset. All addresses within the IOCL are assumed to be logical and map block boundary crossings need not be resolved.

Physical IOCL address – This is a physical, doubleword address that points to a user-supplied IOCL for SIO format I/O operations. The execute channel program entry point (H.IOCS,10) must be used and bit 2 of word 2 of the FCB is set. All addresses within the IOCL are assumed to be physical and all map block boundary crossings are assumed to be resolved.

Word 9

This word specifies the number of bytes of data to be transferred.

Word 10

For nonexecute channel program format, this word defines a device command.

Word 11

Reserved — should be set to zero.

Word 12

This word contains status sent from the user's device or if bit 29 of word 3 is set, this word defines the opcode processor (EP5) detected errors as follows:

Value	Explanation
1	request made with non-expanded FCB
2	FCB format transfer count was zero
3	FCB format, byte transfer count was not a multiple of 4 bytes
4	SIO format with a physical IOCL request by an unprivileged caller
5	SIO format with a physical IOCL request by a nonresident caller
6	first IOCB in caller's IOCL is a transfer in channel
7	caller's IOCL not on a doubleword boundary
8	SIO format IOCL contains an IOCB with a zero transfer count
9	infinite transfer in channel loop
10	consecutive SOBNZ's in IOCL

11	SOBNZ target is not in the IOCL
12	the transfer address is not on a word boundary
13	unprivileged caller's input buffer includes protected locations
14	unprivileged caller's input buffer is unmapped either in MPX-32 or below DSECT
15	cyclic I/O request was made for which no cyclic IOQ is current
16	cyclic I/O request was made and permanent IOQ support was not sysgened into the system

Word 13

Bits 0-7 reserved

Bits 8-31 contain the address of the user-supplied routine to branch to for no-wait I/O normal completion. This routine must be terminated by calling H.IOCS,34 (no-wait I/O end action return). If word 2 bit 12 is reset, this address plus one word is the location where control is transferred on asynchronous status notification.

Word 14

Bits 0-7 reserved

Bits 8-31 contain the address of the user-supplied routine to branch to for no-wait I/O error completion. This routine must be terminated by calling H.IOCS,34 (no-wait I/O end action return).

Word 15

Reserved — should be set to zero.

L.8 File Pointer Table (FPT)

The file pointer table (FPT) provides the linkage between the file control block (FCB) and the file assignment table (FAT). It also allows for multiple logical file code assignments to be made equivalent to the same FAT. The linkage to the FAT is performed at assignment. The linkage to the FCB is performed at open and is re-established if necessary for every operation at opcode processing time. The FPT resides in the task's service area.

FPT entries one to six are reserved for the system as follows:

Entry 1 - System LFC *s*
Entry 2 - Load module LFC *LM
Entry 3 - H.VOMM resource descriptor LFC (1)
Entry 4 - H.VOMM directory LFC (2)
Entry 5 - H.VOMM DMAP/SMAP LFC (3)
Entry 6 - H.VOMM modify resource descriptor LFC X'FFFEE'

Each FPT entry has the following format:

	0 7	8 15	16 23	24 31
Word 0	Reserved	Logical file code (FPT.LFC)		
1	Flags (FPT.FLGS). See Note 1.	FCB address (FPT.FCBA)		
2	Reserved	FAT address (FPT.FATA)		

Notes:

1. Bits in FPT.FLGS are assigned as follows:

Bit	Meaning if Set
0	reserved
1	multiple FPT entries exist that point to the same FAT (i.e., $ASSIGN4 or $ASSIGN *lfc* TO LFC = *lfc* statements)
2	FPT busy flag
3	FPT open
4	this FPT entry is not in use
5	pseudo-SYC assignment (used by TSM)
6	pseudo-FPT for unassigned tempory file
7	reserved

L.9 Parameter Task Activation Block

The following is the structure of the expanded parameter task activation block:

Byte	Word	0 7	8 15	16 23	24 31
0	0	PTA.FLAG	PTA.NRRS	PTA.ALLO	PTA.MEMS
4	1	PTA.NBUF	PTA.NFIL	PTA.PRIO	PTA.SEGS
8	2-3	PTA.NAME			
10	4-5	PTA.PSN			
18	6-7	PTA.ON			
20	8-9	PTA.PROJ			
28	10	PTA.VAT	PTA.FLG2	PTA.EXTD	
2C	11	PTA.PGOW			
30	12	PTA.TSW			
34	13	PTA.RPTR			
38	14	PTA.PGO2			
3C	15	PTA.FSIZ		PTA.RSIZ	
40	16-19	Reserved (zero)			
50-*nn*	20-*nn*	RRS List			

Byte (Hex)	Symbol	Description
0	PTA.FLAG	contains the following:

Bit	Contents
0	reserved
1	job oriented (PTA.JOB)
2	terminal task (PTA.TERM)
3	batch task (PTA.BTCH)
4	debug overlay required (PTA.DOLY)
5	resident (PTA.RESD)
6	directive file active (PTA.DFIL)
7	SLO assigned to SYC (PTA.SLO)

For unprivileged callers, bits 0-3 are not applicable. These characteristics are inherited from the parent task.

1	PTA.NRRS	number of resource requirements or zero if same as summary entries in the load module or executable image preamble

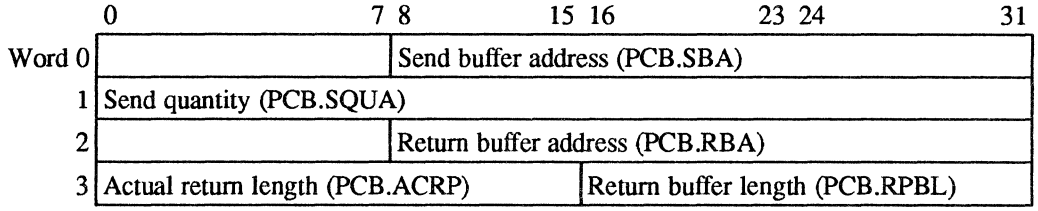
Byte (Hex)	Symbol	Description
2	PTA.ALLO	memory requirement: number of 512-word pages exclusive of TSA, or zero if same as the preamble
3	PTA.MEMS	memory class (ASCII E, H or S) or zero if memory class is to be taken from the preamble. If the memory class is to be taken from the preamble, the caller has the option of specifying the task's logical address space in this field as follows: Bits — Contents 0-3 — hexadecimal value 0 through F representing the task's logical address space in megabytes where zero is 1MB and F is 16MB 4-7 — zero
4	PTA.NBUF	the number of blocking buffers required or zero if same as the preamble
5	PTA.NFIL	the number of FAT/FPT pairs to be reserved or zero if same as the preamble
6	PTA.PRIO	the priority level at which the task is to be activated or zero for the cataloged load module priority. See the Parameter Send Block section in Chapter 2 of the MPX-32 Reference Manual Volume I, for more details.
7	PTA.SEGS	the segment definition count or reserved (zero)
8	PTA.NAME	contains the load module or executable image name, left justified and blank filled, or word 2 is zero and word 3 contains a pathname vector or RID vector
10	PTA.PSN	contains the 1- to 8-character ASCII pseudonym, left justified and blank filled, to be associated with the task or zero if no pseudonym is desired. For unprivileged callers, this attribute is inherited from the parent task if zero is supplied or the parent is in a terminal or batch job environment.
18	PTA.ON	contains the 1- to 8-character ASCII owner name, left-justified and blank-filled, to be associated with the task or zero if the task to default to the current owner name. Valid only when task has system administrator attribute.
20	PTA.PROJ	contains the 1- to 8-character ASCII project name, left-justified and blank-filled, to be associated with files referenced by this task, or zero if same as LMIT
28	PTA.VAT	the number of volume assignment table (VAT) entries to reserve for dynamic mount requests or zero if same as the preamble

Byte (Hex)	Symbol	Description
29	PTA.FLG2	contains the following flags: Bit — Meaning if Set 0 — debug activating task (PTA.DBUG) 1 — Command Line Recall and Edit is in effect for the task (PTA.CLRE) 2 — NOTSA option (PTA.NTSA) 3 — TSA option (PTA.TSA) 4 — expanded PTASK block flag (must be set to use options 33-64) (PTA.EBLK) 5 — reserved (zero) 6 — enables NOMAPOUT option (PTA.NMAP) 7 — enables MAPOUT option (PTA.MAP)
2A	PTA.EXTD	contains the following values: Bit — Meaning if Set -1 — maxaddr of extended MPX-32 and TSA -2 — minaddr of extended MPX-32 and TSA 0 — invalid with PTA.TSA or PTA.NTSA option *n* — a positive number representing a map block of MPX-32 and TSA
2C	PTA.PGOW	contains the initial value of the task option word or zero
30	PTA.TSW	contains the initial value of the task status word or zero
34	PTA.RPTR	contains a pointer to the resource requirement summary list or, if an expanded PTASK block is not used, the RRS list begins here (see resource requirement summary list description below)
38	PTA.PGO2	contains the initial value of the second task option word
3C	PTA.FSIZ	contains the length of the fixed portion of the PTASK block in bytes
3E	PTA.RSIZ	contains the number of bytes of the resource requirement summary
40	Reserved	

Byte (Hex)	Symbol	Description
50		resource requirement summary list. Each entry contains a variable length RRS. The RRS list has up to 384 words. Each entry must be doubleword bounded. Each entry is compared with the RRS entries in the LMIT. If the logical file code currently exists, the specified LFC assignment will override the cataloged assignment, otherwise the special assignment will be treated as an additional requirement and merged into the list. If MPX-32 Revision 1.*x* format of the RRS is specified, it is converted to the format acceptable for assignment processing by the Resource Management Module (H.REMM). See MPX-32 Revision 1.*x* Technical Manual for format of the RRS.

L.10 TSM Procedure Call Block (PCB)

The PCB contains the information necessary for the service to complete a procedure call. The format of the PCB is as follows:



Send buffer address	is the address of a character string that represents a valid TSM procedure call directive
Send quantity	contains the length in bytes of the TSM procedure call directive
Return buffer address	is the address of a buffer to contain either valid return information or an error message if CC1 is set and R7 contains a value of 1
Actual return length	is the number of bytes returned from the procedure call
Return buffer length	is the size in bytes of the supplied return buffer

L.11 Pathname Blocks (PNB)

The pathname block (PNB) is an alternative form of a pathname that can be used interchangeably with pathnames. Because of its structure, it can be parsed faster than a pathname. The PNB is a doubleword bounded, variable length ASCII character string which H.VOMM can distinguish from a pathname since the PNB always starts with an exclamation point.

H.VOMM provides a service to convert a pathname to a PNB. The examples which follow illustrate common pathnames and their corresponding PNB.

Example 1

`@VOL1(DIR1)FILE1`

Word 0	! V O L
1	blank
2	V O L 1
3	blank
4	blank
5	blank
6	! D I R
7	R O O T
8	D I R 1
9	blank
10	blank
11	blank
12	! R E S
13	blank
14	F I L E
15	1 ⌀ ⌀ ⌀
16	blank
17	blank

Example 2

`FILE1`

Word 0	! V O L
1	W O R K
2	! D I R
3	W O R K
4	! R E S
5	blank
6	F I L E
7	1 ⌀ ⌀ ⌀
8	blank
9	blank

Example 3

(DIRECTORY)MYFILE

Word 0	! V O L
1	W O R K
2	! D I R
3	R O O T
4	D I R E
5	C T O R
6	Y ∅ ∅ ∅
7	blank
8	! R E S
9	blank
10	M Y F I
11	L E ∅ ∅
12	blank
13	blank

Example 4

@SYSTEM(SYSTEM)LOADMOD

Word 0	! V O L
1	S Y S T
2	! D I R
3	S Y S T
4	! R E S
5	blank
6	L O A D
7	M O D ∅
8	blank
9	blank

L.12 Post Program-Controlled Interrupt Notification Packet (PPCI)

If a task sets up a PPCI end-action receiver to check status during execution of its channel program, the status is returned in a notification packet. The address of the notification packet is contained in register three upon entering the task's PPCI end-action receiver. The notification packet is described below.

	0 7	8 15	16 23	24 31
Word 0	String forward address (NOT.SFA)			
1	String backward address (NOT.SBA)			
2	Link priority (NOT.PRI)	NOT.TYPE See Note 1.	Reserved	
3	FCB address (NOT.CODE)			
4	PSD 1 of task's PPCI receiver (NOT.PSD1)			
5	PSD 2 of task's PPCI receiver (NOT.PSD2)			
6	Number of PPCIs received since last buffer clear (NOT.STAR)		Number of status doublewords in status buffer (NOT.STAS)	
7	Address of PPCI status buffer (NOT.STAA)			
8	Address of buffer storing next status doubleword (NOT.STPT)			
9	Reserved			
10-*n*	PPCI status buffer			

Notes:

1. NOT.TYPE - Set to 1 for asynchronous notification.
2. Words 0-9 are updated by the operating system and must not be changed by the user.

L.13 Parameter Receive Block (PRB)

The parameter receive block (PRB) is used to control the storage of passed parameters into the receiver buffer of the destination task. The same format PRB is used for message and run requests. The address of the PRB must be presented when the M.GMSGP or M.GRUNP services are invoked by the receiving task.

	0 7	8 15	16 23	24 31
Word 0	Status (PRB.ST)	Parameter receiver buffer address (PRB.RBA)		
1	Receiver buffer length (PRB.RBL)		Number of bytes actually received (PRB.ARQ)	
2	Owner name of sending task, word one (PRB.OWN)			
3	Owner name of sending task, word two			
4	Task number of sending task (PRB.TSKN)			

Notes:

1. Status (PRS.ST) contains the status-value encoded status byte:

Code	Definition
0	normal status
1	invalid PRB address (PRB.ER01)
2	invalid receiver buffer address or size detected during parameter validation (PRB.RBAE)
3	no active send request (PRB.NSRE)
4	receiver buffer length exceeded (PRB.RBLE)

2. Parameter receiver buffer address (PRB.RBA) contains the word address of the buffer where the sent parameters are stored.
3. Receiver buffer length (PRB.RBL) contains the length of the receiver buffer (0 to 768 bytes).
4. Number of bytes received (PRB.ARQ) is set by the operating system and is clamped to a maximum equal to the receiver buffer length.
5. Owner name of sending task (PRB.OWN) is a doubleword that is set by the operating system to contain the owner name of the task that issued the parameter send request.
6. Task number of sending task (PRB.TSKN) is set by the operating system to contain the task activation sequence number of the task that issued the parameter send request.

L.14 Parameter Send Block (PSB)

The parameter send block (PSB) describes a send request issued from one task to another. The same PSB format is used for both message and run requests. The address of the PSB (word bounded) must be specified when invoking the M.SMSGR or M.SRUNR services, but is optional when invoking the M.PTSK service.

When a load module name is supplied in words 0 and 1 of the PSB, the operating system searches the system directory only. For activations in directories other than the system directory, a pathname or RID vector must be supplied.

When activating a task with the M.SRUNR or M.PTSK service, the value specified in byte 0 of PSB word 2 (PSB.PRI) is used to determine the task's execution priority. This value overrides the cataloged priorities of the sending and receiving tasks and the priority specified in the PTASK block. However, priority clamping is used to prevent time-distribution tasks from using this value to execute at a real-time priority, and real-time tasks from executing at a time-distribution priority. Values that can be specified in PSB.PRI are 1-64 (to be the task priority), zero (to use the base priority of the sending task), and X'FF' (to ignore the PSB priority field).

A PSB can be specified as a parameter for the M.PTSK service, along with the required task activation (PTASK) block. The PTASK block also contains a priority specification field. The PSB priority value always overrides the PTASK block priority value.

	0 7	8 15	16 23	24 31
Word 0	Load module or executable image name (PSB.LMN) or zero if activation (or task number (PSB.TSKN) if message or run request to multicopied task)			
1	Load module or executable image name, pathname vector, or RID vector if activation (or zero if message or run request to multicopied task)			
2	Priority (PSB.PRI)	Reserved	Number of bytes to be sent (PSB.SQUA)	
3	Reserved	Send buffer address (PSB.SBA)		
4	Return parameter buffer length in bytes (PSB.RPBL)		Number of bytes actually returned (PSB.ACRP)	
5	Reserved	Return parameter buffer address (PSB.RBA)		
6	Reserved	No-wait request end action address (PSB.EAA)		
7	Completion status (PSB.CST)	Processing start status (PSB.IST)	User status (PSB.UST)	Options (PSB.OPT)

Word 0

Bits 0-31 — Load module or executable image name — contains characters 1 through 4 of the name of the load module or executable image to receive the run request or

Task number — contains the task number of the task to receive the message or the task number of the multicopied load module or executable image to receive the run request.

Word 1

Bits 0-31 — Load module or executable image name — contains characters 5 through 8 of the name of the load module or executable image to receive the run request, or zero if the message or run request is sent to multicopied load module or executable image.

Word 2

Bits 0-7 — Contains the priority at which the receiver task is expected to be activated. Valid values are 1-64, zero, (for base priority of the sending task) and X'FF', which generates activation priority based on a combination of values that can be specified during task activation.

The following tables show how the priority of a receiver task is determined when activated with M.SRUNR or with M.PTSK.

When Activating with M.SRUNR

Send Task	Cataloged Priority of Receive task	Priority in PSB	Activates Receive task at
1-54	1-54	0	Send task cat. priority
1-54	55-64	0	55 (time-dist. clamp)
55-64	1-54	0	54 (real-time clamp)
55-64	55-64	0	Send task cat. priority
*	1-54	1-54	PSB priority
*	1-54	55-64	54 (real-time clamp)
*	55-64	1-54	55 (time-dist. clamp)
*	55-64	55-64	PSB priority
*	*	X'FF'	Receive task cat. priority

* not specified

When Activating with M.PTSK

Send Task	Cataloged Priority of Receive task	Priority in PTASK block	PSB	Activates Receive task at
1-54	1-54	0	0	Send task cat. priority
1-54	55-64	0	0	55 (time-dist. clamp)
1-54	*	1-54	0	Send task cat. priority
1-54	*	55-64	0	55 (time-dist. clamp)
55-64	1-54	0	0	54 (real-time clamp)
55-64	55-64	0	0	Send task cat. priority
55-64	*	1-54	0	54(real-time clamp)
55-54	*	55-64	0	Send task cat. priority
*	1-54	0	1-54	PSB priority
*	1-54	0	55-64	54 (real-time clamp)
*	55-64	0	1-54	55 (time-dist.clamp)
*	55-64	0	55-64	PSB priority
*	*	1-54	1-54	PSB priority
*	*	1-54	55-64	54 (real-time clamp)
*	*	1-54	X'FF'	PTASK block priority
*	*	55-64	1-54	55 (real-time clamp)
*	*	55-64	55-64	PSB priority
*	*	55-64	X'FF'	PTASK block priority
*	*	0	X'FF'	Receive task cat. priority

* not specified

Bits 8-15 reserved

Bits 16-31 Number of bytes to be sent — specifies the number of bytes to be passed (0 to 768) with the message or run request.

Word 3

Bits 0-7 reserved

Bits 8-31 Send buffer address — contains the word address of the buffer containing the parameters to be sent.

Word 4

Bits 0-15 Return parameter buffer length — contains the maximum number of bytes (0 to 768) that may be accepted as returned parameters.

Bits 16-31 Number of bytes actually returned — set by the send message or run request service upon completion of the request.

Word 5

Bits 0-7 reserved

Bits 8-31 Return parameter buffer address — contains the word address of the buffer where any returned parameters are stored.

Word 6

Bits 0-7 reserved

Bits 8-31 No-wait request end-action address — contains the address of a user routine to be executed at a software interrupt level upon completion of the request.

Word 7

Bits 0-7 Completion status — contains completion status information posted by the operating system as follows:

Bit	Meaning if Set
0	operation in progress (PSB.OIP)
1	destination task was aborted before completion of processing for this request (PSB.DTA)
2	destination task was deleted before completion of processing for this task (PSB.DTD)
3	return parameters truncated — attempted return exceeds return parameter buffer length (PSB.RPT)
4	send parameters truncated — attempted send exceeds destination task receiver buffer length (PSB.SPT)
5	user end-action routine not executed because of task abort outstanding for this task (can be examined in abort receiver to determine incomplete operation) (PSB.EANP)
6-7	reserved

Bits 8-15 Processing start (initial) status — contains initial status information posted by the operating system as follows:

Bit	Meaning if Set
0	normal initial status (PSB.IST)
1	message request task number invalid (PSB.TSKE)
2	run request load module or executable image name not found (PSB.LMNE)
3	reserved
4	file associated with run request load module or executable image name does not have a valid load module or executable image format (PSB.LMFE)
5	dispatch queue entry (DQE) space is unavailable for activation of the load module or executable image specified by a run request (PSB.DQEE)
6	an I/O error was encountered while reading the directory to obtain the file definition of the load module or executable image specified in a run request (PSB.SMIO)
7	an I/O error was encountered while reading the file containing the load module or executable image specified in a run request (PSB.LMIO)
8	memory unavailable
9	invalid task number for run request to module or executable image in RUNW state
10	invalid priority specification. An unprivileged task can not specify a priority which is higher than its own execution priority (PSB.PRIE).
11	invalid send buffer address or size (PSB.SBAE)
12	invalid return buffer address or size (PSB.RBAE)
13	invalid no-wait mode end action routine address (PSB.EAE)
14	memory pool unavailable (PSB.MPE)
15	destination task receiver queue is full (PSB.DTQF)

Bits 16-23 User status — defined by the destination task.

Bits 24-31 Options — contains user-request control specification as follows:

Bit	Meaning if Set
24	request is to be issued in no-wait mode (PSB.NWM)
25	do not post completion status or accept return parameters. This bit is examined only if bit 24 is set. When this bit is set, the request was issued in the no call back mode. (PSB.NCBM).

L.15 Resource Create Block (RCB)

Each H.VOMM entry point that creates a permanent file, a temporary file, a memory partition, or a directory may receive a resource create block (RCB) in order to fully define the attributes of the resource that is created. RCB formats are described in the next three tables. RCBs must be doubleword bounded.

If an RCB is not supplied by the caller, the resource is created with the default attributes described in the MPX-32 Reference Manual Volume I, Chapter 4.

Permanent and Temporary File Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0	File owner name (RCB.OWNR)
1	
2	File project group name (RCB.USER)
3	
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8	Maximum extension increment (RCB.MXEI). See Note 3.
9	Minimum extension increment (RCB.MNEI). See Note 4.
10	Maximum file size (RCB.MXSZ). See Note 5.
11	Original file size (RCB.OSIZ). See Note 6.
12	File starting address (RCB.ADDR). See Note 7.
13	File RID buffer (RCB.FAST). See Note 8.
14	Option flags (RCB.OPTS). See Note 9.
15	Default override (RCB.FREE). See Note 10.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
2	modify access allowed (RCB.MODI)
3	update access allowed (RCB.UPDA)
4	append access allowed (RCB.APPN)
9	delete access allowed (RCB.DELE)

2. Resource management flags. For any bit not set, system defaults apply and, in some cases, the default is the equivalent of the bit being set (optional):

Bit	Description
0-7	resource type, equivalent to file type code, interpreted as two hexadecimal digits, 0 - FF (RCB.FTYP)
8-10	reserved
11	file EOF management required (RCB.EOFM)
12	fast access (RCB.FSTF)
13	do not save (RCB.NSAV)
14	reserved for MPX-32 usage
15	file start block requested (RCB.SREQ)
16	file is executable (RCB.EXEC)
17	owner ID set on access (RCB.OWID)
18	project group ID set on access (RCB.UGID)
19	reserved
20	maximum file extension increment is zero. System default value not used. (RCB.MXEF)
21	minimum file extension increment is zero. System default value not used (RCB.MNEF)
22	reserved
23	zero file on creation/extension (RCB.ZERO)
24	file automatically extendible (RCB.AUTO)
25	file manually extendible (RCB.MANU)
26	file contiguity desired (RCB.CONT)
27	shareable (RCB.SHAR) (owner rights spec only)
28	link access (RCB.LINK)
29-30	reserved
31	file data initially recorded as blocked (RCB.BLOK)

3. Maximum extension increment is the desired file extension increment specified in blocks (optional). Default is 64 blocks.
4. Minimum extension increment is the minimum acceptable file extension increment specified in blocks (optional). Default is 32 blocks.
5. Maximum file size is the maximum extendible size for a file specified in blocks (optional).
6. Original file size is the original file size specified in blocks (optional). Default is 16 blocks.

7. File starting address is the disk block where the file should start, if possible. If the space needed is currently allocated, an error is returned (optional).
8. File RID buffer is the address within the file creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created file is not returned to the creating task.
9. Option flags bits are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)

10. Default override - If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bit	Description
0-7	must be zero
8-10	reserved
11	file EOF management not required
12	fast access not required
13	resource can be saved
14-22	reserved
23	do not zero file on creation/extension
24	file is not automatically extendible
25	file is not manually extendible
26	file contiguity is not desired
27	resource is not shareable
28-30	reserved
31	file data initially recorded as unblocked

Directory Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0-1	Directory owner name (RCB.OWNR)
2-3	Directory project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-10	Reserved
11	Directory original size (RCB.OSIZ). See Note 3.
12	Directory starting address (RCB.ADDR). See Note 4.
13	Directory RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications bits are as follows:

Bit	Description
0	read access allowed (RCB.READ)
8	directory may be traversed (RCB.TRAV)
9	directory may be deleted (RCB.DELE)
10	directory entries may be deleted (RCB.DEEN)
11	directory entries may be added (RCB.ADEN)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)
27	shareable (RCB.SHAR)

3. Directory original size is the number of entries required (optional).
4. Directory starting address is the disk block number where the directory should start, if possible. If the space needed is currently allocated, an error is returned (optional).
5. Directory RID buffer is the address within the directory creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created directory is not returned to the creating task.

6. Option flags are as follows:

Bit	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)

7. If default override is set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional).

Bit	Description
0-7	must be zero
13	resource can be saved
27	resource is not shareable

Memory Partition Resource Create Block (RCB)

	0 7 8 15 16 23 24 31
Word 0-1	Partition owner name (RCB.OWNR)
2-3	Partition project group name (RCB.USER)
4	Owner rights specifications (RCB.OWRI). See Note 1.
5	Project group rights specifications (RCB.UGRI). See Note 1.
6	Other's rights specifications (RCB.OTRI). See Note 1.
7	Resource management flags (RCB.SFLG). See Note 2.
8-9	Reserved
10	Starting word page number (RCB.PPAG)
11	Partition original size (RCB.OSIZ). See Note 3.
12	Partition starting address (RCB.ADDR). See Note 4.
13	Partition RID buffer (RCB.FAST). See Note 5.
14	Option flags (RCB.OPTS). See Note 6.
15	Default override (RCB.FREE). See Note 7.

Notes:

1. Rights specifications are optional:

Bit	Description
0	read access allowed (RCB.READ)
1	write access allowed (RCB.WRIT)
9	delete access allowed (RCB.DELE)

2. Resource management flags are optional:

Bit	Description
13	do not save (RCB.NSAV)

3. Partition's original size is the number of protection granules required.
4. Partition's starting address is a 512-word protection granule number in the user's logical address space where the partition is to begin.
5. Partition's RID buffer is the address within the partition creator's task where the eight word resource identifier (RID) is to be returned. If this parameter is not supplied (i.e., is zero), the RID for the created partition is not returned to the creating task.
6. Option flags are optional:

Bits	Description
0	owner has no access rights (RCB.OWNA)
1	project group has no access rights (RCB.USNA)
2	others have no access rights (RCB.OTNA)
9	defines a static partition (RCB.STAT)
24-31	define memory class (RCB.MCLA). Values are:

Value	Memory Class
0	S (default)
1	E
2	H
3	S

7. If set, these bits override any corresponding bit set in RCB.SFLG and the system defaults (optional):

Bits	Description
0-7	must be zero
13	resource can be saved

L.16 Resource Identifiers (RID)

The fastest means of locating a volume resource (once created) is by its resource identifier (must be on a doubleword boundary). The resource identifier has the following format:

	0 7	8 15	16 23	24 31
Word 0-3	Volume name			
4	Creation date			
5	Creation time			
6	Volume address of resource descriptor			
7	Must contain zero		Resource type	

Since the resource identifier contains the volume address of the resource descriptor, the resource descriptor (which points to and describes the resource) can be accessed directly without going through the various directories which would otherwise have to be traversed.

Given a valid pathname defining a resource, the corresponding resource descriptor may be retrieved by the H.VOMM locate resource service. The first eight words of a resource descriptor consist of the resource identifier.

L.17 Resource Logging Block (RLB)

The resource logging block (RLB) is a word-bounded data structure used to pass information between H.VOMM and the caller. The information is used to locate a directory entry and resource descriptor for a single resource or for all resources defined in a particular directory.

	0 7	8 15	16 23	24 31
Word 0	Pathname vector or RID address (RLB.TGT)			
1	Resource directory buffer address (192W) (RLB.BUFA). See Note 1.			
2	Associated mounted volume table entry address (RLB.MVTE)			
3	Parent directory RD block address (RLB.RDAD)			
4	Type (RLB.TYPE). See Note 2.	Buffer offset (RLB.BOFF)		
5	Length. See Note 3.	Return buffer address (RLB.DIRA)		
6	User FCB address (RLB.FCB)			
7	Flags. See Note 4.	Reserved (RLB.INT)		

Notes:

1. Optional. If not specified, a resource directory is not returned.
2. Bits in RLB.TYPE are assigned as follows:

Bits	Meaning if Set
0	indicates recall (RLB.RECA)
1-7	reserved

3. This word contains the address of a buffer and its length in words (the buffer can be up to 16 words long).
4. Bits in the flags byte are assigned as follows:

Bits	Meaning if Set
0-1	reserved
2	directory entry and resource descriptor for specified directory are returned
3	root directory
4	resource is located
5-7	reserved

L.18 Resource Requirement Summary (RRS) Entries

The resource requirement summary (RRS) is a doubleword bounded data structure used to identify the resources required by a task to the resource manager. Resources are statically allocated using the information in the RRS entry. The RRS is generally built by processors requiring static allocation of resources, such as TSM, cataloger, etc., or supplied as an argument for dynamic allocation.

For compatibility purposes, revision 1.*x* RRS formats can be used. The details of these formats can be found in Chapter 2 of a revision 1.*x* Technical Manual.

Type 1 - Assign by Pathname

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Plength (RR.PLEN)	Reserved. See Note 2.
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-*n*	Pathname (variable length) (RR.NAME1)			

Type 2 - Assign to Temporary File

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Initial file size (RR.PLEN)	
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1) (Volume name is optional)			

Type 3 - Assign to Device

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Density (RR.DENS). See Note 5.	Zero
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Device type (RR.DT3). See Note 6.	Volume number (RR.VLNUM)	Channel number See Note 7. (RR.CHN3)	Subchannel number (RR.SCHN3)
5	Unformatted ID (1-4 characters) (RR.UNFID)			

Type 4 - Assign to LFC

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Zero	Logical file code (RR.SFC)		
3	Options (RR.OPTS). See Note 4.			

Type 5 - Assign by Segment Definition

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	UDT index (RR.UDTI)	Reserved
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Starting block number (RR.STBLK)			
5	Number of blocks (RR.NBLKS)			

Type 6 - Assign by Resource ID

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	Reserved
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)			
8	Binary creation date (RR.DATE)			
9	Binary creation time (RR.TIME)			
10	Resource descriptor block address (RR.DOFF)			
11	Reserved		Resource type (RR.RTYPE)	

Type 7 - Reserved for Future Use

Type 8 - Reserved for Future Use

Type 9 - Mount by Device Mnemonic

	0 7	8 15	16 23	24 31
Word 0	Zero	System ID (RR.SYSID). See Note 11.		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Zero	
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4-7	Volume name (16 characters; left-justified, blank-filled) (RR.NAME1)			
8	Device type (RR.DT9). See Note 8.	Reserved	Channel number (RR.CHN9). See Note 9.	Subchannel number (RR.SCHN9)
9	Zero			

Type 10 - Assign to ANSI Tape

	0 7	8 15	16 23	24 31
Word 0	Zero	Logical file code (RR.LFC)		
1	Type (RR.TYPE). See Note 1.	Size (RR.SIZE)	Format (RR.FORM)	Protect (RR.PROT)
2	Access (RR.ACCS). See Note 3.			
3	Options (RR.OPTS). See Note 4.			
4	Record length (RR.RECL)		Block size (RR.BSIZE)	
5	Generation number (RR.GENN)			
6	Generation version number (RR.GENV)			
7	Absolute termination date (RR.EXPIA)			
8	Relative termination date (RR.EXPIR)		Logical volume identifier (RR.LVID)	
9	RR.LVID (cont.)			
10-13	17-character file identifier (RR.AFID)			
14	RR.AFID (cont.)	Reserved		
15	Reserved			

Type 11 - Assign to Shadow Memory

	0 7	8 15	16 23	24 31
Word 0	Zero			
1	Type (RR.TYPE) See Note 1.	Size (RR.SIZE)	Shadow flags (RR.SHAD). See Note 10.	
2	Start address (RR.SADD)			
3	End address (RR.EADD)			

Notes:

1. Bits in RR.TYPE are assigned as follows:

Value	Meaning
1	assign by pathname (RR.PATH)
2	assign to temporary file (RR.TEMP)
3	assign to device (RR.DEVC)
4	assign to secondary LFC (RR.LFC2)
5	assign to segment definition (RR.SPACE)
6	assign by resource ID (RR.RID)
7	reserved for future use
8	reserved for future use
9	mount by device mnemonic (RR.MTDEV)
10	assign to ANSI labeled tape (RR.ANS)
11	assign to shadow memory (RR.SHTYP)
12-255	reserved

2. Byte 3 is zero. This field is used by MPX-32 for big blocking buffers.
3. Bits in RR.ACCS are assigned as follows:

Bits	Meaning if Set
0	read access allowed (RR.READ)
1	write access allowed (RR.WRITE)
2	modify access allowed (RR.MODFY)(not valid for ANSI tapes)
3	update access allowed (RR.UPDAT)
4	append access allowed (RR.APPND)
5-15	reserved
16	explicit shared use requested (RR.SHAR)
17	exclusive use requested (RR.EXCL)
18	assign as volume mount device (RR.MNT)
19-31	reserved

4. Bits in RR.OPTS are assigned as follows:

Bits	Meaning if Set
0	treat as SYC file (RR.SYC) (TSM/JOB only)
1	treat as SGO file (RR.SGO) (TSM/JOB only)
2	treat as SLO file (RR.SLO)
3	treat as SBO file (RR.SBO)
4	explicit blocked option (RR.BLK)
5	explicit unblocked option (RR.UNBLK)
6	inhibit mount message (RR.NOMSG)
7	reserved for system use
8	automatic open requested (RR.OPEN)
9	user-supplied blocking buffer address in FCB (RR.BUFF)
10-11	reserved for system use
12	mount with no-wait (RR.NOWT)
13	mount as public volume (RR.PUBLC)
14	set by H.VOMM for special case handling of VOMM assignments (RR.VOMM)
15	file is spooled when deallocated (RR.SEP)
16	ANSI labeled tape on RRS type 3 (RR.ANSI)
17-31	reserved

5. RR.DENS contains the density specification for XIO high speed tape units. When specified, this field has the following bit significance:

Bits	Meaning if Set
0	indicates 800 bpi nonreturn to zero inverted (NRZI)
1	indicates 1600 bpi phase encoded (PE)
6	indicates 6250 bpi group coded recording (GCR)

If this field is zero, 6250 BPI is set by default.

6. RR.DT3 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	channel present
1-7	device type

7. RR.CHN3 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	subchannel is present. Examined only if bit zero of RR.DT3 is set.
1-7	channel number

8. RR.DT9 specifies whether or not a channel is present and specifies the device type:

Bits	Meaning if Set
0	channel present
1-7	device type

9. RR.CHN9 specifies whether or not a subchannel is present and specifies the channel number:

Bits	Meaning if Set
0	subchannel is present. Examined only if RR.DT9 is set.
1-7	channel number

10. RR.SHAD contains the shadow flags that qualify the start and end addresses, or specify what portions of the task are to be shadowed:

Bits	Meaning if Set
0-7	reserved
8	shadow the task (RR.SHTSK)
9	shadow the TSA (RR.SHTSA)
10	shadow the stack (RR.SHST)
11	shadow memory is required (RR.SHRQ)
12	shadow the entire task (RR.SHALL)
13	absolute address (RR.ABS)
14	relative to the code section origin (RR.CREL)
15	relative to the data section origin (RR.DREL)

11. RR.SYSID is the ID for mounting a multiprocessor volume. Valid IDs are:

    Multiported (MP) 0 through F

    Dual Ported (DP) 0 or 1

For more information on mounting multiprocessor volumes see the MPX-32 Reference Manual Volume I, Chapter 4, Mounting Multiprocessor Volumes.

L.19 Receiver Exit Block (RXB)

The receiver exit block (RXB) is used to control the return of parameters and status from the destination (receiving) task to the task that issued the send request. It is also used to specify receiver exit options. The same format RXB is used for both messages and run requests. The address of the RXB must be presented as an argument when either the M.XMSGR or M.XRUNR services are called.

	0 7	8 15	16 23	24 31
Word 0	Return status (RXB.ST)	Return parameter buffer address (RXB.RBA)		
1	Options (RXB.OPT)	Reserved	Number of bytes to be returned (RXB.RQ)	

Notes:

1. Return status (RXB.ST) contains status as defined by the receiver task. Used to set the user status byte in the parameter send block (PSB) of the task which issued the send request.
2. Return parameter buffer address (RXB.RBA) contains the word address of the buffer containing the parameters which are to be returned to the task which issued the send request.
3. Options (RXB.OPT) contains receiver exit control options. It is encoded as follows:

Value	Exit Type	Meaning
0	M.XRUNR	wait for next run request.
	M.XMSGR	return to point of task interrupt.
1	M.XRUNR	exit task, process any additional run requests. If none exist, perform a standard exit.
	M.XMSGR	N/A

4. Number of bytes to be returned (RXB.PQ) contains the number of bytes (0 to 768) of information to be returned to the sending task.

L.20 Type Control Parameter Block (TCPB)

The type control parameter block (TCPB) allows I/O to and from the system console by setting up task buffer areas for messages output by a task and optional reads back from the console. If no input is desired, word one of the TCPB must be zero.

See the MPX-32 Reference Manual Volume I, Chapter 5 for further details on the TCPB.

	0 — 11	12	13 — 31
Word 0	Output quantity (TCP.OQ)	See Note 1.	Output data address (TCP.OTCW)
1	Input quantity (TCP.IQ)	See Note 1.	Input data address (TCP.ITCW)
2	Console Teletype Flags (TCP.FLGS). See Note 2.		

Notes:

1. Bit 12 is set to 1.
2. Bits in TCP.FLGS are assigned as follows:

Bits	Meaning if Set
0	no-wait I/O
31	operation in progress. This bit is reset after post-I/O processing completes.

Type Control Parameter Block (TCPB) using 24-bit address:

	0 7	8 15	16 23	24 31
Word 0	Output quantity (TCP.OQ)	Output data buffer address (TCP.OTCW)		
1	Input quantity (TCP.IQ)	Input data buffer address (TCP.ITCW)		
2	Console device flags (TCP.FLGS) See Note 1.			

Notes:

1. Bit interpretations for TCP.FLGS are:

Bits	Meaning if Set
0	no-wait I/O
1	data buffer addresses are 24-bit addresses (TCP.LAD) Note: This bit must be set.
31	operation in progress. This bit is reset after post-I/O processing completes.

L.21 Unit Definition Table (UDT)

The unit definition table (UDT) is a system resident structure that identifies device-dependent information required by a handler for a specific device. The UDT is built by the SYSGEN process, one for each device configured in the system. During SYSGEN, each UDT is linked to its corresponding controller definition table (CDT) and its associated controller and handler.

	0 7	8 15	16 23	24 31
Word 0	UDT index (UDT.UDTI)		CDT index (UDT.CDTI)	
1	Unit status (UDT.STAT). See Note 1.	Device type code (UDT.DTC). See Note 2.	Logical channel number (UDT.CHAN)	Logical subaddress (UDT.SUBA)
2	Reserved	Address of dispatch queue entry of task which has device allocated if device is not shared (UDT.DQEA)		
3	Physical channel number (UDT.PCHN)	Physical subaddress (UDT.PSUB)	Sectors per block (UDT.SPB) or number of characters per line (UDT.CHAR). See Note 3.	Sectors per allocation unit (UDT.SPAU) or number of lines per screen (UDT.LINE). See Note 4.
4	Flags (UDT.FLGS). See Note 5.	Number of sectors per track on disk or global line counter if a terminal (UDT.SPT)	Maximum byte transfer (UDT.MBX)	
5	Number of sectors on disk or tab setting if a terminal (UDT.SECS)			
6	Sector size, on disk or a tab setting if a terminal (UDT.SSIZ)		Number of heads on disk or a tab setting if a terminal (UDT.NHDS)	
7	Serial number if tape or removable disk (UDT.SERN). See Note 6.			
8	Peripheral time-out value (UDT.PTOV)			
9	Reserved	Address of device context area (UDT.DCAA) or handler name at initialization (UDT.HNAM)		
10	Bit flags (UDT.BIT2). See Note 7.		Associated allocated resource table index if assigned (UDT.ARTI)	
11	Service interrupt handler address (UDT.SIHA)			
12	Reserved (UDT.CXR) See Note 8.	Secondary flags (UDT.BIT3) See Note 9. or UDT.HIST. See Note 10	Reserved (UDT.SHFL)	Reserved (UDT.DQEN)
13	Address of first IOQ linked to this device (UDT.FIOQ)			
14	Address of last IOQ linked to this device (UDT.BIOQ)			
15	Link Priority (UDT.LPR1)	Link Count (UDT.IOCT)	Unit Status byte 2 (UDT.STA2). See Note 11.	

Notes:

1. Bits in UDT.STAT are assigned as follows:

Bit	Meaning if Set
0	online (UDT.ONLI)
1	dual-portd XIO disk (UDT.DPDC)
2	allocated (UDT.ALOC)
3	terminal in use and not in wait (UDT.USE)
4	system output unable to allocate (UDT.NOAL)
5	shared device (UDT.SHR)
6	premounted (UDT.PREM)
7	terminal (TSM) device (UDT.TSM)

2. For example, 01 for any disk, 04 for any tape, etc. Valid device type codes are listed in Appendix A.
3. For disks, contains the number of sectors per block (UDT.SPB). For terminals, contains the number of characters per line (UDT.CHAR).
4. For disks, contains the number of sectors per allocation unit (UDT.SPAU). For SLO or terminals, contains the number of lines per page or screen (UDT.LINE).
5. Bits in UDT.FLGS are assigned as follows:

Bit	Meaning if Set
0	extended I/O device (UDT.FCLS)
1	I/O outstanding (UDT.IOUT)
2	removable disk pack (UDT.RMDV)
3	a break has been requested for this device (UDT.LOGO)
4	autoselectable for batch SLO (UDT.BSLO)
5	autoselectable for batch SBO (UDT.BSBO)
6	autoselectable for real-time SLO (UDT.RSLO)
7	autoselectable for real-time SBO (UDT.RSBO)

6. If the device is a terminal or console, the first halfword is the current terminal type for TERMDEF (UDT.CTDF) and the second halfword is the default terminal type (UDT.DTDF).
7. Bits in UDT.BIT2 are assigned as follows:

Bits	Meaning if Set
0	port is private; else switched (UDT.DIAL)
1	port is connected to modem (UDT.MODM)
2	port has graphic capability (UDT.GRFC)
3	port is full duplex (UDT.FDUX)
4	port is configured multidrop (UDT.MDRA)
5	volume mounted on device (UDT.VOL)
6	echo by computer (UDT.ECHO)
7	device has failed. Log off TSM (UDT.DEAD)
8	cache device (UDT.CAC)
9	inhibit automatic line wrap (UDT.NRAP)
10	spool device requires form feed after printing rather than before; initial form feed is inhibited (UDT.FEOP)

Bits	Meaning if Set
11	quarter inch cartridge tape drive (UDT.QITD)
12	software read flow control required (UDT.RXON)
13	software write flow control required (UDT.WXON)
14	hardware read flow control required (UDT.RHWF)
15	hardware write flow control required (UDT.WHWF)

8. For switched port, contains the value specified in the LOGONFLE CXR = option (UDT.CXR)
9. Bits in UDT.BIT3 are assigned as follows:

Bits	Meaning if Set
0	SCSI device (UDT.SCSI)
1-7	reserved

10. UDT.HIST is used as an address save area by pseudo device handlers, such as ON.IPXIO
11. Bits in UDT.STA2 are assigned as follows:

Bits	Meaning if Set
0	IOQ linked from UDT (UDT.IOQ)
1	IOP device (initialized by SYSGEN) (UDT.IOP)
2	device malfunction (UDT.MALF)
3	operator intervention applicable (UDT.INTV)
4	use standard XIO interface
5	floppy disk
6	cartridge module drive
7	moving head disk with fixed head option
8	if software read flow control enabled, use DTR line; otherwise, use RTS line. (UDT.RDTR)
9	memory disk (UDT.MD) or valid command line recall and edit device (UDT.CLRE)
10	memory allocated for memory disk (UDT.MDAL)
11	start address of memory disk specified at SYSGEN (UDT.MDST)
12	multiport device is shared with an MPX-32 Revision 3.2C or earlier version (UDT.PPV)
13	device is exclusive ANSI (UDT.ANSI)
14	serial printer (UDT.SLPR)
15	port is switched and CXR=N option has been specified (UDT.DCXR)

Glossary

access method	A software package that provides the ability to access fields within records, to classify or order records according to the contents of fields, and to perform other such functions.
access mode	Defines the range of operations to be performed on a resource.
aged page	A page which has not been referenced within a predetermined frame of time during demand page processing. This page is no longer considered a part of the task's working set.
allocated resource table (ART)	A system resident table with an entry for each currently allocated resource in the system.
allocation	The process of securing a resource for a specific usage and access mode for a task.
allocation unit	A mechanism for grouping more than one block on a formatted disc, or other mass medium, at one time. Usually specified in multiples of 192-word disc blocks. See disc block.
argument	A value (string or integer) that is assigned to a parameter.
assign	To associate a resource with a logical file code used by a process.
assignment	The process of associating a logical file code with a system resource. Does not guarantee the resource for a specific use or access mode for a task.
asynchronous	Implies one entity does not wait for or otherwise acknowledge another entity when it performs an operation.
asynchronous notification	A process does not stop execution waiting for notification. It receives a software interrupt when an asynchronous operation is complete.
base mode	Implies the base register instruction set that allows executable programs of up to 4096KW (16MB).
blocked I/O	The process of packing records equal to or less than 254 bytes so that more than one record is stored in a 192-word disc block.
blocking buffers	Buffers used for packing records for blocked I/O. See blocked I/O.

caller notification packet (CNP)
: A structure used to supply additional calling parameters and to control the handling of abnormal conditions that may occur during resource requests.

classes of users
: A three-level grouping of users into OWNER, PROJECTGROUP and OTHER. Used to permit or limit access to a resource by 'class'.

command file
: A file containing commands known to a particular operating system or process.

CONCEPT/32
: A term which implies the entire line of CONCEPT/32 computers; for example, the 32/67.

configuration
: Hardware: the physical hardware related to a CPU. Software: adapting the operating system to a hardware configuration with the SYSGEN processor.

data files
: Files containing data or transactions that have been processed or will be processed by a task.

data management
: The ability to structure data into records using buffers.

Datapool
: An area of memory that contains the same functionality as Global Common but with the added flexibility of symbolic references being independent of the actual positioning of data within the memory area. See Global Common.

deallocate
: To detach a resource from a process.

deassign
: To remove the association between a logical file code and a resource and deallocate the resource.

dequeue
: To remove from a prioritized list.

demand page
: Allocation of memory when the logical page is referenced by the task on demand. The process of allocating physical memory when pages are referenced and deallocating physical memory when pages are no longer active. Pages that are no longer active are considered aged and removed from the task's working set.

device
: A peripheral unit such as a card reader, a printer, a disc drive, or a tape drive. Distinguished from media used with devices.

device access
: Levels are physical I/O, logical device I/O, and logical file I/O.

device-dependent I/O
: Tasks perform operations to a specified device with minimal IOCS overhead.

device-independent I/O
: Tasks perform I/O operations through the use of operating system calls which are independent of the device used to perform the operation.

direct I/O	Tasks perform operations bypassing IOCS and handler functions by coding its own handler and attaching it to a specific channel.
directory	A list of file names and/or memory partition names. Stored on disc like a regular file. Located via a resource descriptor for the directory. Directory names are 1 to 16 characters in length and valid characters for names are A to Z, 0 to 9, dot (.) and underscore (_).
directory descriptor	The resource descriptor for a directory.
disc block	A common unit of measurement (some number of words) used to measure file space on formatted media throughout a system. The number of words in a block is oriented to the most common sector size on discs used with the system.
DMAP	See resource descriptor allocation map.
dynamic assignment	The association of a logical file code with a system resource during task execution.
enqueue	To put into a list ordered by software priority.
exclusive use	A resource is not available for use by any other task until that resource is deallocated by the using task. Guarantees access to a resource, within the access limitations imposed by the resource creator, when logical I/O is initiated.
executable image	A file of object code produced by the LINKER/X32.
explicit shared use	A resource can be used concurrently by more than one task. Each task maintains resource integrity by establishing its own synchronization and locking mechanisms. Each task is guaranteed access to the resource, within access limitations imposed by the resource creator, when logical I/O is initiated.
extended code	That part of the operating system that has been modified to run in the extended execution space.
extended file control block	A file control block set up by the user which contains more information than the standard file control block. See file control block.
file	A set of information stored on a mass medium such as disc or tape that is given a unique identity (number and often name) and treated as a single entity for processing.
file control block (FCB)	Set up by the user to describe logical files within a task. Describes attributes of logical I/O operation.
file descriptor	A resource descriptor for a file.

file identifier	A unique identifier stored in the resource descriptor for a file when the file is created. Used to access the resource descriptor without a directory search.
file segment	Set of contiguous allocation units on a volume identifying the space associated with a file. Each file segment definition contains the absolute 192-word block volume segment address and the segment length in 192-word blocks (maximum of 32 file segment definitions per file).
file space allocation map (SMAP)	A bit map used to allocate space on a volume.
filename	A 1- to 16-character name supplied for a permanent file when it is created on a mass medium. Used in most cases thereafter to identify the file. Valid characters for filenames are the upper-case letters A to Z, the decimal numbers 0 to 9, and the special characters dot (.) and underscore (_). Filenames to be used with the compatible interfaces, for example Editor, File Manager, and Media, are limited to 1 to 8 characters.
format	Standard organization of information.
formatted volume	A disc pack or floppy disc that contains standard volume system structures established by the Volume Formatter utility.
Global Common	An area of memory accessible by using symbolic names to identify specific storage cells. Programs belonging to many independent tasks can freely access the same data and exchange control information within the Global Common area.
implicit shared use	A resource is available for concurrent use by other tasks in a compatible access mode. Does not guarantee access when logical I/O is initiated. Resource integrity is automatically maintained by the system.
job file	A command file designed to run in the batch or interactive environment.
library file	Object modules or source modules identified by name that are output to a single file. Modules on library files can be used separately and repeatedly. For example, object modules can be retrieved by name during cataloging and inserted with existing code. The ability to edit the contents of library files by name is also normally available.
load module file	A file of object code produced by the Cataloger that is ready to relocate from disc into memory and execute as a process. Load module files can be activated by name and are controlled by name or task number.

logical device I/O	I/O where the physical characteristics of a device are not determined automatically by the file management system (device and data formatting are inhibited), allowing the user to exert control over a particular physical device or device medium.
logical dismount	The action taken by MPX-32 to disassociate a volume from the requesting task. A TSM logical dismount disassociates the volume from the requesting context.
logical file code (LFC)	User defined 1- to 3-character ASCII codes identifying logical files within tasks.
logical file I/O	I/O where the physical characteristics of a device and device medium (device format control, data conversion, data formatting) are performed automatically for the user so that he gains a degree of device independence.
logical mount	The action taken by MPX-32 to associate a physically mounted volume to a task. A TSM logical mount associates the volume to the TSM context requesting the mount.
logical resource	Any entity existing only because of a mechanism provided by software. The primary logical resources are: disc volumes, directories, files, and memory partitions.
map block	A 2048-word unit of memory allocation. In demand page processing, a page is a map block.
medium (singular) media (plural	A contiguous source of input or output that is used for a particular peripheral device. For example, a disc pack is the medium mounted on a disc drive; a tape is the medium mounted on a tape drive; paper is the medium used on a printer; a deck of cards is the medium used on a card reader. The operating system distinguishes use of media from use of devices.
memory descriptor	The resource descriptor for a memory partition.
memory partitions	Named areas of physical memory that can be shared by concurrently executing tasks.
modular	Construction in independent layers. Each higher level layer builds on the layer beneath it and provides its own standard interfaces to the levels above and below it.
mounted volume table (MVT)	A system resident table with an entry for each physically mounted volume. Each entry contains information used by the system to maintain volume accounting information.

multicopied tasks — Tasks with the same name and the same concurrent load module activity, owned by a single owner or several owners. This is accomplished by cataloging a task as multicopy. Task numbers must be used to communicate with multicopied tasks. See task number.

multiprocessor volume — A specially mounted user volume that allows tasks operating in separate system environments to concurrently access any volume resource.

multivolume magnetic tape — A set of 1 through 255 maximum physical reels of magnetic tape processed as a continuous reel.

nonbase mode — Implies the nonbase register instruction set which allows executable programs of up to 128KW.

nonpublic volume — A volume assigned specifically to the tasks that mount it. Remains physically mounted until use and assign counts equal 0.

object file — A file of assembled or compiled code that can be cataloged or linked into a task.

owner — The user who has possession of and can control access to a file, device, memory partition, or directory. Usually the owner of a resource is the user who created its resource descriptor.

owner name — A 1- to 8-character name supplied at logon which remains unchangeable through logoff. The following characters cannot be used in owner names: blanks, commas, semicolons, equal signs, line feeds, dollar signs, percent signs, exclamation points, and left or right parentheses. All other characters are valid. Owner names are associated with any task or process activated on the system and noted by any process that acts in the owner's behalf. Owner name is also associated with any resources a user creates unless the user specifies otherwise. Specifying a different owner when creating a resource definition does not change the user's owner name; it only specifies the owner name associated with the resource.

page — A 512-word unit of memory protection. Also referred to as a protection granule. Four pages compose a map block.

For demand page processing, a page is a map block brought into memory and removed from memory during the life of a demand page task.

page fault — The reference of a page within the logical address space which is not currently a part of the task's working set.

page in — Bringing into logical memory a page needed to satisfy an address referenced by a task.

page out — The removal of aged pages from the task's logical address space.

parameter — A symbolic name in a process or directive file that can be assigned an argument.

pathname — Variable length ASCII character strings which uniquely identify a volume resident resource by explicitly or implicitly describing the volume, one or more directories, and the resource name.

pathname block — Doubleword bounded variable length ASCII character string beginning with "!" which uniquely identifies a volume resident resource by explicitly or implicitly describing the volume, one or more directories, and the resource name.

permanent files — Files that remain defined on a volume until explicitly deleted.

physical dismount — The action taken by MPX-32 to disassociate a volume from an assigned mount device and deallocate the device.

physical mount — The action taken by MPX-32 to allocate a mount device and associate that device to the assigned volume name.

physical resource — Any physical hardware that MPX-32 supports. Tasks access the resource to perform their functions. The primary physical resources are: the CPU, computer memory (main storage), and input/output devices.

portable — Can be used on any compatible device in a single system configuration. Can also be carried to a compatible device on a different system hardware configuration. Usually describes a volume.

post program-controlled interrupt receiver — User supplied end-action receiver entered when a hardware post program-controlled interrupt is encountered.

process — A body of code scheduled for CPU time as a single entity. A load module is a process, in loadable form, stored on disc. Same as task.

project group name — A name that is specified at logon and can also be changed. Identifies a group of users that have a defined set of rights when they access a resource.

protect — To limit access to a resource. See classes of users.

protection granule — A 512-word unit of memory protection. Also referred to as a page in a non-demand page context. Four protection granules compose a map block.

public volume — A volume available for resource assignments by all tasks activated in the system.

real time task — Synonymous with time critical process.

requestor — The process which requests a function. Each process on a system has an associated owner name. The system process that requests a function for a user (e.g., in the interactive environment) keeps track of the owner name so that the user thinks of himself as the 'requestor'.

resource — Any source of support that exists external to a task and that the task needs to perform its function. A resource can be physical or logical.

resource create block (RCB)
Defines access attributes for permanent files, temporary files, memory partitions, and directories when the particular resource is created. If not supplied at resource creation, system default attributes are assumed.

resource descriptor (RD)
Contains access, accounting, and space definition information pertaining to mounted volume resources, permanent files, temporary files, directories and partitions.

resource descriptor allocation map (DMAP)
A bit map used for the allocation of resource descriptors on a volume.

resource identifier (RID)
The fastest way to locate an already created volume resource. The RID is in the first eight words of a resource descriptor and contains the volume address of the resource descriptor, which points to and describes the resource.

resource logging block (RLB)
A parameter block used as input to the M.LOGR service for logging resources.

Resource Management Module (H.REMM)
Performs allocation and assignment of all system resources and maintains access compatibility and usage rights for these resources. Also contains synchronization mechanisms for concurrent access to shared resources.

resource requirement summary (RRS)
Defines assignment requirements of a resource. Entries are variable length, doubleword bounded. There are 9 types of entries.

root directory — The directory of all directories defined on a volume.

SMAP — See file space allocation map.

source file — A file of source code to be assembled or compiled into object code.

static assignment — The association of a logical file code with a system resource during task activation.

status posting — The process of returning information that indicates whether a service was completed successfully, with errors, or denied.

swap volume — A volume used as the primary medium for swap file allocations.

symbolic — A representation of a physical resource, e.g., a name that represents an entity but is not the entity itself.

synchronous notification — A process waits on further processing until it is notified that an operation is done or that there is something inhibiting the operation (e.g., a resource is not available or other processes are in contention for the resource).

system administrator attribute (SA) — Gives an unprivileged user the ability to execute privileged SVC's, allows a user to mount public volumes, and allows a user to change his owner name. A user with the system administrator attribute is, however, restricted to resource access limitations imposed by the resource creator.

system directory — Special directory on the system volume which contains volume resources necessary for system operation.

system volume — A volume containing the system and bootstrap images from which the current system was IPLed. This volume is automatically mounted by the SYSINIT task at system initialization.

task — Synonymous with process.

task name — The name supplied when a task is cataloged or linked.

task number — An 8-digit hexadecimal number assigned to a task by MPX-32 when the task is activated. The task number is unique and identifies a particular copy or sharer of a task.

temporary files — Unnamed files that are referenced by resource identifiers. They are automatically deleted from the system and their volume space made available when the last task assigned to them terminates execution.

time critical process — A process which has time constraints. Same as a real time task.

traverse — To pass through a directory on the way to another directory or resource.

type control parameter block (TCPB)
: Set up by the user for sending and receiving messages to/from the system console.

unformatted media
: A medium (magnetic tape, disc pack or floppy disc) that does not contain valid volume format information, but must be mounted before initiation of I/O operations.

usage mode
: Defines the degree to which multiple tasks can concurrently allocate a resource. Usage modes are: exclusive use, explicit shared, and implicit shared.

user
: A person who uses a system. Processes and commands that activate processes are either initiated by a user or initiated on behalf of a user.

volume
: A medium that has a standard format. Disc packs can be formatted as volumes.

volume assignment table (VAT)
: A task resident table with an entry for each non-public volume currently assigned to the task.

Volume Management Module (H.VOMM)
: Manipulates volume resident and related memory resident structures in order to allow for creation, deletion, and maintenance of user and system resources which reside on volumes; for example, provides space management for all currently mounted volumes in the system.

working set
: The pages (map blocks) of a task that are actively being referenced within a predetermined frame of time.

Index

- A -

Abort a Task, (V1)2-46, (V1)6-16, (V1)6-50, (V1)7-21, (V1)7-59, (V2)2-6
Abort Codes, C-1
  display description, (V2)1-58
  system files, (V3)10-18
Abort Receiver, (V1)2-34, (V1)6-166, (V1)7-170
Abort Self Service, (V1)6-17, (V1)7-22
Abort Specified Task Service, (V1)6-16, (V1)7-21
Abort with Extended Message Service, (V1)6-18, (V1)7-23
ACM/MFP
  controller record
    defaults, (V3)10-33
    syntax, (V3)10-33
  initialization format, (V3)10-35
  set dual-channel mode, (V1)6-142, (V1)7-145
  set single-channel mode, (V1)6-145, (V1)7-148
  true full-duplex operation, (V3)10-37
Acquire Current Date/Time in ASCII Format Service, (V1)6-138, (V1)7-139
Acquire Current Date/Time in Binary Format Service, (V1)6-20, (V1)7-25
Acquire Current Date/Time in Byte Binary Format Service, (V1)6-15, (V1)7-20
Acquire System Date/Time in Any Format Service, (V1)6-85, (V1)7-88
ACS
  CONCEPT 32/67 usage, (V3)12-2
  description, (V3)12-1
  directive summary, (V3)12-5
  DUMPACS
    description, (V3)12-5
    directive summary, (V3)12-5
    directives
      CHECKSUM, (V3)12-6
      COMPARE, (V3)12-6
      DUMP, (V3)12-8
      EXIT, (V3)12-9
      MODE, (V3)12-10
      REVISION, (V3)12-12
    errors, (V3)12-16
  firmware file, (V3)12-2
  LOADACS
    description, (V3)12-4
    directive file (M.ACS), (V3)12-1
    directive summary, (V3)12-5
    directives
      COPY, (V3)12-7
      ENABLE, (V3)12-8
      LOAD, (V3)12-9
      PATCH, (V3)12-11
      VERIFY, (V3)12-13
    errors, (V3)12-14
  M.ACS, (V3)12-1
    sample file, (V3)12-13
Activate a Task, see Task, execution
Activate Job, (V1)6-226
  also see Task, execution
Activate Load Modules, (V3)7-10, (V3)7-50
Activate Program at Given Time-of-Day Service, (V1)6-185, (V1)7-187
Activate Task Interrupt Service, (V1)6-97, (V1)7-101
Activate Task Service, (V1)6-5, (V1)7-7
Add
  new users to the system, (V3)10-4
  project group names to the system, (V3)10-12
Advance Record or File Service, (V1)6-74, (V1)7-9
AIDDB, (V1)1-16
ALIM
  initialization format, (V3)10-26
  terminal record
    defaults, (V3)10-25
    syntax, (V3)10-25
Allocate, resource, (V1)5-1, (V1)5-2, (V1)5-3, (V1)6-8, (V1)7-12
Allocate File or Peripheral Device Service, (V1)6-222
Allocate File Space Service, (V1)6-206
Allocate Resource Descriptor Service, (V1)6-208
Allocated Resource Table (ART)
  display, (V4)2-15
  specify size, (V3)7-11
Allocation Unit, (V1)5-55
Alterable Control Store, see ACS
Analyze, system tables and queues, (V4)2-3
ANSI Labeled Tapes
  assign
    LFC, (V2)1-38
    RRS, (V1)5-12
  dismount, (V2)7-8
  display, (V2)7-10, (V2)7-11
  examples, (V2)7-6

exclude support, (V3)7-40
file records
fixed-length, (V2)7-2
spanned, (V2)7-3
variable-length, (V2)7-3
implementation levels, (V2)7-5
interchange with other systems, (V2)7-4
labels, (V2)7-4
LVID, (V2)7-1
messages, (V2)7-5
mount, (V2)7-9
overview, (V2)7-1
tape drives, (V2)7-3
usage, (V2)7-2
utilities
ADMOUNT, (V2)7-8
AMOUNT, (V2)7-9
ASTAT, (V2)7-10
AVOLM, (V2)7-11
J.LABEL, (V2)7-13, (V3)10-62
VID, (V2)7-2
write header labels, (V2)7-13
Archive, floppy disk, (V4)2-42
Arithmetic Exception Handling, (V1)2-34
Arithmetic Exception Inquiry Service, (V1)6-182, (V1)7-184
ASCII Date/Time to Binary Conversion, (V1)6-28, (V1)7-34
ASCII Decimal to Binary Conversion, (V1)6-26, (V1)7-32
ASCII Files, display two, (V4)2-24
ASCII Hex to Binary Conversion, (V1)6-27, (V1)7-33
ASCII Interchange Code Set, F-1
ASMX32, (V1)1-17
ASSEMBLE, (V1)1-16
Assembler/X32, (V1)1-17
Assembly Source Code Flowchart Tool, (V4)2-28
Assign
arithmetic result to a parameter, (V2)1-96
integer value to a parameter, (V2)1-96
logical file codes, (V2)1-36
resource, (V1)5-2, (V1)5-3, (V1)6-8, (V1)7-12
string value to a parameter, (V2)1-94
Assign and Allocate Resource Service, (V1)6-8, (V1)7-12
Asynchronous Task Interrupt Service, (V1)6-10, (V1)7-14
Automatic Batch Job Submission on Boot-up, (V4)2-15
Automatic IPL, (V3)6-4
Automatic Mounting of Public Volumes, (V1)4-20, (V3)9-11

- B -

Backspace Record or File Service, (V1)6-11, (V1)7-16
Backspace magnetic tape, (V2)3-22
Bad Blocks, (V1)5-64
Base Mode
exclude support, (V3)7-40
nonshared tasks, (V1)2-3
system services, (V1)7-1
Batch Environment
accessing, (V2)1-12
example, (V2)1-109
Batch Job Entry Service, (V1)6-13, (V1)7-18
Batch Processing
activate job, (V1)6-13, (V1)7-18 (V2)1-45, (V2)1-59, (V2)1-102, (V2)2-8
change job priority, (V2)1-103
continuous processing, (V2)2-31, (V3)7-36
example, (V2)1-109, (V2)1-110
maximum number of active jobs, (V3)7-32
overview, (V1)1-14
specify priority level, (V3)7-11
terminate input stream, (V2)1-106
Batch Stream Memory Pool Interaction, (V2)1-115
Binary Date/Time to ASCII Conversion, (V1)6-30, (V1)7-36
Binary Date/Time to Byte Binary Conversion, (V1)6-33, (V1)7-39
Binary to ASCII Decimal Conversion, (V1)6-29, (V1)7-35
Binary to ASCII Hexadecimal Conversion, (V1)6-31, (V1)7-37
Blocked I/O, (V1)3-4, (V1)3-15, (V1)5-34
Blocking Buffers, (V1)3-15
Boot Block, (V1)4-22
Booting the System
from Master SDT, (V3)2-12
control switches, (V3)2-21, K-1
from User SDT, (V3)4-4
Bootstrap program, (V3)5-1
Bootstrapping, (V3)5-1
philosophy, (V3)2-22
Branch
backward, (V2)1-62
conditional, (V2)1-64, (V2)1-66, (V2)1-69
forward, (V2)1-63
Break Key, (V1)6-60, (V1)6-65, (V1)7-68, (V1)7-71, (V2)1-19
Break/Task Interrupt Link/Unlink Service, (V1)6-19, (V1)7-24

Building SYSGEN Input File
COMPRESS task, (V3)3-2
directive input file, (V3)3-1
object input file, (V3)3-1
Byte Binary Date/Time to ASCII Conversion, (V1)6-32, (V1)7-38
Byte Binary Date/Time to Binary Conversion, (V1)6-34, (V1)7-40

- C -

C.TRACE, (V3)7-64
Caller Notification Packet, see CNP
Carriage Control Characters, L-31
Case Sensitivity, TSM, (V2)1-20
CATALOG
overlays, (V1)1-16
privilege, (V1)1-15
CDOT array, specify size, (V3)7-12
CDT, L-3
Central Processing Unit, see CPU
Change
current working directory, (V1)6-46, (V1)7-56, (V2)1-48, (V2)1-104
default system input device, (V2)2-56
default system output device, (V2)2-56
directories, (V2)1-11
key, (V2)1-9
owner attributes, (V3)10-7
password, (V2)1-10
project group, (V1)6-46, (V1)7-56, (V2)1-11, (V2)1-48
project group name key, (V3)10-13
SBO device, (V2)2-37
SLO device, (V2)2-37
tabs, (V4)2-56
task priority, (V1)6-131, (V1)7-131
Change Defaults Service, (V1)6-46, V1)7-56
Change Priority Level Service, (V1)6-131, (V1)7-131
Change Task to Unprivileged Mode Service, (V1)6-192, (V1)7-192
Change Temporary File to Permanent File Service, (V1)6-175, (V1)6-248, (V1)7-177
Channel Configuration, (V3)7-13
Channel Reservation, (V1)6-153, (V1)7-156
Channel Reservation Release, (V1)6-149, (V1)7-153
Channel Status, display, (V2)2-46
Check TERMDEF Additions, (V4)2-19
Clear
break receiver, (V1)6-19, (V1)7-24
M.KEY file, (V3)10-9
M.PRJCT file, (V3)10-14
options, (V2)1-81
TSM directives, (V2)1-49
Close File Service, (V1)6-23, (V1)7-29
Close Resource Service, (V1)6-21, (V1)7-27
CNP
description, (V1)5-15
PPCI receiver, (V1)5-44, (V1)5-47, L-53
return conventions, (V1)5-16
status posting, (V1)5-16, (V1)5-58
structure, L-2
Command Line, (V1)6-24, (V1)7-30, (V2)1-85, (V2)10-1
Command Line Recall and Edit
disable, (V2)10-8
edit, (V2)10-2
enable, (V2)10-8
introduction, (V2)10-1
MPX.PRO file, (V2)10-6
recall, (V2)10-4
Common Area, see Memory Partition
Communicating
with another task, (V2)1-14, (V2)2-22, (V2)2-42
with other terminals, (V2)1-14
Communication
internal, (V1)1-12
intertask, (V1)1-11, (V1)2-22, (V2)2-42
Communications Facilities, (V1)1-11
Compare Program Source Files, (V4)2-25
Compatibility Mode Services
exclude support, (V3)7-40, (V3)7-61
COMPRESS
accessing, (V2)4-1
at SYSGEN, (V3)7-1
description, (V2)4-1, (V3)3-2
error messages, (V2)4-3
example, (V2)4-4
logical file codes, (V2)4-2, (V2)4-3
Compressed Source Format, I-1
CONCEPT/32
interrupts and traps, (V1)1-5
machine type, (V3)7-34
Conditional Branch, (V2)1-64, (V2)1-66, (V2)1-69
Conditional Processing, (V2)1-29
Configuration Module, (V4)2-18
Connect Task to Interrupt Level, (V2)2-10
Connect Task to Interrupt Service, (V1)6-35, (V1)7-41
Console
configuration, (V3)2-1, (V3)7-34
device definition, (V3)7-21
Context Switch Timing for M.SURE, (V4)2-20
Continue Task Execution, (V2)1-20, (V2)2-11
Control Switches, (V3)2-21, K-1

Controller Definition Table (CDT), L-3
Controller Status, display, (V2)2-46
Convert
ASCII date/time to binary, (V1)6-28, (V1)7-34
ASCII decimal to binary, (V1)6-26, (V1)7-32
ASCII hex to binary, (V1)6-27, (V1)7-33
binary date/time to ASCII, (V1)6-30, (V1)7-36
binary date/time to byte binary, (V1)6-33, (V1)7-39
binary to ASCII decimal, (V1)6-29, (V1)7-35
binary to ASCII hexadecimal, (V1)6-31, (V1)7-37
byte binary date/time to ASCII, (V1)6-32, (V1)7-38
byte binary date/time to binary, (V1)6-34, (V1)7-40
date and time formats, (V1)7-43
pathname to pathname block, (V1)6-129, (V1)7-129
Convert ASCII Date/Time to Byte Binary Format Service, (V1)6-25, (V1)7-31
Convert ASCII Date/Time to Standard Binary Service, (V1)6-28, (V1)7-34
Convert ASCII Decimal to Binary Service, (V1)6-26, (V1)7-32
Convert ASCII Hexadecimal to Binary Service, (V1)6-27, (V1)7-33
Convert Binary Date/Time to ASCII Format Service, (V1)6-30, (V1)7-36
Convert Binary Date/Time to Byte Binary Service, (V1)6-33, (V1)7-39
Convert Binary to ASCII Decimal Service, (V1)6-29, (V1)7-35
Convert Binary to ASCII Hexadecimal Service, (V1)6-31, (V1)7-37
Convert Byte Binary Date/Time to ASCII Service, (V1)6-32, (V1)7-38
Convert Byte Binary Date/Time to Binary Service, (V1)6-34, (V1)7-40
Convert Pathname to Pathname Block Service, (V1)6-129, (V1)7-129
Convert System Date/Time Format Service, (V1)6-39, (V1)7-50
Convert Tape to MPX-32 2.*x*, (V2)3-23
Convert Time Service, (V1)7-43
Copy a File, (V2)3-25
CPU
dispatch queue area, (V1)2-42
execution of IPU tasks, (V1)2-16
scheduling, (V1)1-7, (V1)2-10
execution priorities, (V1)2-10
real-time priority levels, (V1)2-10
state chain management, (V1)2-12
time-distribution priority levels, (V1)2-11
CPU Execution Time
display, (V2)1-99
for the task, (V1)6-204, (V1)7-206
Crash Codes, (V1)2-51, C-32
Crash Dump Analyzer, (V4)2-3
Create
directory, (V1)4-26, (V1)6-53, (V1)7-61, (V2)3-32
FCB, (V1)5-39, (V1)7-45
file, (V1)4-32, (V1)6-37, (V1)6-228, (V1)7-46, (V2)1-50, (V2)3-34
memory partition, (V1)4-43, (V1)6-108, (V1)7-111, (V2)3-29
shared image, (V1)4-45
temporary file, (V1)4-41, (V1)6-173, (V1)6-209, (V1)7-48
timer entry, (V1)6-159, (V1)7-163
timer table, (V2)2-58
Create Directory Service, (V1)6-53, (V1)7-61
Create File Control Block Service, (V1)7-45
Create Memory Partition Service, (V1)6-108, (V1)7-111
Create Permanent File Service, (V1)6-37, (V1)6-228, (V1)7-46
Create Temporary File Service, (V1)6-173, (V1)6-209, (V1)7-48
Create Timer Entry Service, (V1)6-159, (V1)7-163
CSECT, (V1)2-3, (V1)3-21
Current Working Directory
change, (V1)6-46, (V1)7-56, (V2)1-48, (V2)1-104
description, (V1)4-6

- D -

Datapool, (V1)1-12, (V1)3-19, (V1)3-20, (V3)7-39
Datapool Editor, (V1)1-17
Date and Time
conversion
any format, (V1)6-39, (V1)7-43, (V1)7-50
ASCII to binary, (V1)6-28, (V1)7-34
ASCII to byte binary, (V1)6-25, (V1)7-31
binary to ASCII, (V1)6-30, (V1)7-36
binary to byte binary, (V1)6-33, (V1)7-39
byte binary to ASCII, (V1)6-32, (V1)7-38
byte binary to binary, (V1)6-34, (V1)7-40
display, (V2)2-57

formats, H-1
inquiry
any format, (V1)6-85, (V1)7-84, (V1)7-88
in ASCII, (V1)6-44, (V1)6-138, (V1)7-52, (V1)7-139
in binary, (V1)6-20, (V1)7-25
in byte binary, (V1)6-15, (V1)7-20
system update, (V2)2-20
Date and Time Inquiry Service, (V1)6-44, (V1)7-52
Deallocate File or Peripheral Device Service, (V1)6-231
Deallocate File Space Service, (V1)6-211
Deallocate Resource Descriptor Service, (V1)6-212
Deassign and Deallocate Resource Service, (V1)6-42, (V1)7-53
Debug Link Service, (V1)6-123, (V1)7-207
Debugger
execute, (V1)6-45, (V1)7-55
system, see System Debugger
task
AIDDB, (V1)1-16
specify default, (V3)7-16
transfer control to, (V1)6-213, (V1)7-207
unsupported, (V4)2-44
DEBUGX32, (V1)1-18
Decrease File Size, (V2)3-65
Default
SBO device, (V3)7-45
SID device, (V3)7-51
SLO device, (V3)7-63
task debugger, (V3)7-16
user directory, (V3)10-4
user project group, (V3)10-4
user volume, (V3)10-4
Define FCB Macro, (V1)5-39
Define Parameters, (V2)1-51
Delete
directory, (V2)3-38
file, (V1)4-41, (V1)6-232, (V2)1-53, (V2)3-39
memory partition, (V1)4-44, (V1)6-232, (V2)3-37
owner from system, (V3)10-9
project group name from system, (V3)10-13
resources, (V1)6-48, (V1)7-57
SBO file, (V2)2-14
SLO file, (V2)2-13
task, (V1)6-50, (V1)7-59, (V2)1-20
from dispatch queue, (V2)2-25
timer entry, (V1)6-56, (V1)7-66
Delete Permanent File or Non-SYSGEN Memory Partition Service, (V1)6-232
Delete Resource Service, (V1)6-48, (V1)7-57
Delete Task Service, (V1)6-50, (V1)7-59
Delete Timer Entry Service, (V1)6-56, (V1)7-66
Deliverable Software for MPX-32, (V3)2-3
Demand Page
define environment, (V3)7-10, (V3)7-12, (V3)7-17, (V3)7-28
description, (V1)3-17
inhibit support, (V3)7-40
support, (V1)3-17, (V3)7-10, (V3)7-12 (V3)7-17, (V3)7-28
Demonstration
MPX-32, (V4)1-1
TERMDEF, (V4)2-28, (V4)2-31
TSM scanner, (V4)2-59
Detect File Overlap, (V4)2-27
Device
access, (V1)5-21, (V1)5-28, A-1
formatting, (V1)5-47
functions, (V1)5-41, (V1)5-44, L-27, L-30
handlers, (V3)7-15
information display, (V4)2-24
inquiry, (V1)6-246
mnemonics, (V1)6-52, (V1)7-60, A-6
specification, (V1)5-28, A-1
status display, (V2)2-46
type codes, (V1)6-52, (V1)7-60, (V3)7-15, A-6, L-3, L-77
Device Initializer/Loader, see DEVINITL
Device-Dependent I/O, (V1)5-43
Device-Independent I/O, (V1)5-1, (V1)5-32
DEVINITL
activate, (V3)11-6
directive file, (V3)11-1
directive summary, (V3)11-7
directives
DEV_CNTRL, (V3)11-8
DEVICE, (V3)11-9
IDENT, (V3)11-9
OPTION, (V3)11-10
REREAD, (V3)11-11
RETRY, (V3)11-11
WCS_FILE, (V3)11-12
errors, (V3)11-13
example, (V3)11-12
firmware file, (V3)11-3
messages, (V3)11-13
Dial-up Port Protection, (V4)2-43
Direct I/O, (V1)5-34
Directive Files
chaining, (V2)1-27
errors, (V2)1-28, (V2)1-29
examples, (V2)1-111

executing tasks, (V2)1-27
macros, (V2)1-30
nesting, (V2)1-28
read from, (V2)1-89
transfer control, (V2)1-46
Directives, read from a file, (V2)1-89
Directory
access attributes, (V1)3-9
change current working, (V1)6-46, (V1)7-56, (V2)1-48, (V2)1-104
create, (V1)4-26, (V1)6-53, (V1)7-61, (V2)3-32
current working, (V1)4-6
delete, (V2)3-38
description, (V1)4-24
display, (V2)3-44
log information, (V1)6-103, (V1)7-106
protection, (V1)4-24, (V1)4-28
RCB, (V1)5-62, L-63
root, (V1)4-6, (V1)4-26
size allocation, (V2)3-14, (V2)3-15
structure, (V1)4-6, (V1)4-26
system, (V1)5-21
usage, (V1)4-28
user default, (V2)1-11, (V3)10-4
Disable, logons, (V2)1-54
Disable Channel Interrupt (DCI or DI), (V2)2-15
Disable Message Task Interrupt Service, (V1)6-59, (V1)7-67
Disable User Break Interrupt Service, (V1)6-60, (V1)7-68
Disconnect Task from Interrupt Level, (V2)2-15
Disconnect Task from Interrupt Service, (V1)6-55, (V1)7-63
Disk Descriptions, (V1)5-27
Disk Device Codes, (V3)7-21
Disk Dump by File, (V4)2-23
Disk Dump by Sector, (V4)2-22
Disk Error History, display, (V4)2-24
Disk Space Usage, (V4)2-26
Disks, device definitions, (V3)7-19
Dismount a Volume, (V1)4-18, (V1)6-57, (V1)7-64, (V2)1-55, (V2)2-16
Dismount Volume Service, (V1)6-57, (V1)7-64
Dispatch Queue, (V1)2-42
number of entries, (V3)7-26
Dispatch Queue Entry (DQE), L-5
Display
abort code description, (V2)1-58
active SWAPPER time, (V4)2-53
allocated resource table, (V4)2-15
ANSI labeled tape information, (V2)7-10, (V2)7-11
channel status, (V2)2-46
communications region, (V4)2-55
controller status, (V2)2-46
CPU execution time, (V2)1-99
date and time, (V2)2-57
device information, (V4)2-24
device status, (V2)2-46
directories, (V2)3-11, (V2)3-44
disk error history, (V4)2-24
disk space usage, (V4)2-26
file contents, (V2)1-76
file listing, (V2)3-8, (V2)3-9, (V2)3-42, (V2)3-44, (V2)3-46
GPRs, (V1)6-61, (V1)7-69
inswaps, (V4)2-54, (V4)2-55
IPU status, (V2)2-46
IPU traps, (V2)2-26
job accounting file, (V2)1-33, (V2)2-26
job queue, (V2)1-99
load module information, (V4)2-35
logged on users, (V2)1-99, (V2)1-106
MDT, (V4)2-35
memory, (V2)2-46
memory address and contents, (V2)2-41
memory limits, (V1)6-61, (V1)7-69
memory partitions, (V2)3-12, (V2)3-13, (V2)3-44
outswaps, (V4)2-54, (V4)2-55
owners and attributes, (V3)10-9
project group names, (V3)10-13
PSD, (V1)6-61, (V1)7-69
rapid file allocation MDT, (V4)2-35
resources, (V2)3-8, (V2)3-44
saved files, (V2)3-43
shared memory includes, (V4)2-54, (V4)2-55
system configuration, (V4)2-18
system dispatch queue, (V2)2-26
system output queues, (V2)2-26
system patch file, (V2)2-26
tabs, (V4)2-56
task exit status, (V1)7-78
task identification, (V1)6-88, (V1)7-91
task status, (V2)2-26, (V2)2-46
two ASCII files, (V4)2-24
UDT entry, (V4)2-60
volume status, (V2)2-46
word locations, (V2)2-18, (V2)2-45
DMAP, (V1)5-51, (V1)6-208, (V1)6-212
DPEDIT, (V1)1-17
DQE, L-5
DRAM, (V1)3-16, (V3)7-52, (V3)7-54, (V3)7-55
DSECT, (V1)3-21
Dual-Ported Disk
release, (V1)6-142, (V1)7-145
reserve, (V1)6-145, (V1)7-148

Dump, see Display
Dump Disk File, (V4)2-23
Dump System-Configured Disk, (V4)2-22
DUMPACS, see ACS
Duplicate Floppy Disk, (V4)2-42
Dynamic Memory Allocation, (V1)1-8, (V1)3-19

- E -

EDIT, (V1)1-17
Eight-Line Asynch, see ACM
Eight-Line Serial Printer, device definition, (V3)7-21
Eject/Purge Routine Service, (V1)6-214, (V1)7-208
Enable, logons, (V2)1-57
Enable Channel Interrupt (ECI or EI), (V2)2-19
Enable Message Task Interrupt Service, (V1)6-64, (V1)7-70
Enable User Break Interrupt Service, (V1)6-65, (V1)7-71
End Action Wait Service, (V1)6-63, (V1)7-15
End-Action Receivers, (V1)2-22
End-of-Job Designation, (V2)1-58
ENTER CR FOR MORE, (V2)1-21, (V2)1-83, (V3)7-22
Environments, operating, (V2)1-11, (V2)1-13
EOF, write, (V1)6-23, (V1)6-193, (V1)6-196, (V1)6-197, (V1)6-214, (V1)6-215, (V1)7-29, (V1)7-195, (V1)7-198, (V1)7-199, (V1)7-208, (V1)7-209
EOF Management, (V1)3-12, (V1)4-49, (V1)5-36, (V1)5-37
EOM Management, (V1)4-50, (V1)5-36, (V1)5-37
Erase or Punch Trailer Service, (V1)6-215, (V1)7-209
Error Codes, unsupported software, (V4)2-60
Establish a Label, (V2)1-52
Exception Handler, (V1)7-158
Exception Return Address, (V1)7-157
Exclude Memory Partition Service, (V1)6-67
Exclude Shared Image Service, (V1)7-73
Exclusive File Lock
  release, (V1)6-240
  set, (V1)6-241
EXCPM, (V1)5-43, (V1)5-46
Execute a Task, see Task, execution
Execute Channel Program, (V1)5-48, (V1)7-26
Execute Channel Program (EXCPM), (V1)5-43, (V1)5-46
Execute Channel Program File Control Block Service, (V1)7-26
Execute Channel Program Service, (V1)6-216, (V1)7-210
Exit
  J.MDTI, (V2)5-4
  message end-action routine, (V1)6-200, (V1)7-202
  message receiver, (V1)6-201, (V1)7-203
  no-wait I/O end-action routine, (V1)6-199, (V1)7-201
  OPCOM, (V2)2-22
  run receiver, (V1)6-203, (V1)7-205
  run request end-action routine, (V1)6-202, (V1)7-204
  task execution, see Task, execution
  TSM, (V2)1-59
  VOLMGR, (V2)3-40
Exit from Message End-Action Routine Service, (V1)6-200, (V1)7-202
Exit from Message Receiver Service, (V1)6-201, (V1)7-203
Exit from Run Request End-Action Routine Service, (V1)6-202, (V1)7-204
Exit from Task Interrupt Level Service, (V1)6-19, (V1)6-198, (V1)7-24, (V1)7-200
Exit Run Receiver Service, (V1)6-203, (V1)7-205
Exit With Status Service, (V1)7-78
Expand Task's Logical Address Space, (V2)1-101
Extend File Service, (V1)6-70, (V1)7-76
Extended Memory, array, (V1)6-217, (V1)7-211
Extended MPX-32
  aborts and errors, (V1)3-29
  CATALOG, (V1)3-32
  create system, (V1)3-29
  description, (V1)3-23
  designate location, (V1)3-23
  macro assembler, (V1)3-24
  move the non-base TSA, (V2)1-60
  performance, (V1)3-23
  physical memory, (V1)3-25
  program flow control, (V1)3-27
  relocate, (V1)3-31
  resident modules, (V1)3-26, (V3)2-4
  SYSGEN, (V1)3-27, (V1)3-28, (V3)7-28
  task's logical address space, (V1)3-31
  TSM, (V1)3-33, (V2)1-60
Extended TSA, (V1)3-34
Extendibility, (V1)4-30

- F -

Fast Access, (V1)3-13, (V1)4-29
FAT, (V1)5-20, (V1)5-51
Faults, (V1)2-51

FCB
create, (V1)5-48, (V1)7-45
define, (V1)5-39
description, (V1)5-2, (V1)5-38
structure
16-word, L-20
compatible (8-word), L-32
high speed data, L-39
FHD Port
release, (V1)6-219, (V1)7-212
reserve, (V1)6-220, (V1)7-212
File
access attributes, (V1)3-10
access methods, (V1)4-34
access modes, (V1)4-36
address inquiry, (V1)6-234
allocation, (V1)6-222
append mode, (V1)4-39
assign temporary, (V1)4-41, (V1)5-14
assignment, (V1)4-29, (V1)4-33
attachment, (V1)4-29, (V1)4-33
attributes, (V1)4-29
backspace, (V1)6-11, (V1)7-16
block, (V1)4-30
blocked I/O, (V1)3-15
close, (V1)4-40, (V1)6-23, (V1)7-29
contiguous space, (V1)4-30
copy, (V2)3-25
create permanent, (V1)4-32, (V1)6-37, (V1)6-228, (V1)7-46, (V2)1-50, (V2)3-34
create temporary, (V1)4-41, (V1)6-173, (V1)6-209, (V1)7-48
deallocate, (V1)6-231
delete permanent, (V1)4-41, (V1)6-232, (V2)1-53, (V2)3-39
delete temporary, (V1)4-42
description, (V1)4-29
detach permanent, (V1)4-41
detach temporary, (V1)4-42
display contents, (V2)1-76
display listing, (V2)3-42, (V2)3-44, (V2)3-46
EOF management, (V1)3-12
extension
automatic, (V1)3-11, (V1)4-31
contiguous, (V1)3-11
manual, (V1)3-11, (V1)4-31
maximum, (V1)3-12
minimum, (V1)3-12
fast access, (V1)3-13, (V1)4-29, (V1)4-31
granularity, (V1)4-30
increase size, (V2)3-41
log information, (V1)6-103, (V1)6-244, (V1)7-106
management, see File Management
modify mode, (V1)4-38
multiprocessor, (V1)1-9
names, (V1)4-31
no-save, (V1)3-14
open permanent, (V1)4-29, (V1)4-34, (V1)6-236
open temporary, (V1)4-42
operations, (V1)4-34
permanent, (V1)1-8, (V1)4-29, (V1)4-32
positioning
absolute, (V1)4-36
relative, (V1)4-36
print, (V2)1-84
protection, (V1)1-9, (V1)4-32
random access, (V1)1-9, (V1)4-35
rapid allocation, (V2)5-1
RCB, (V1)5-59, L-60
read mode, (V1)4-37
rename, (V1)6-143, (V1)7-146, (V2)1-87, (V2)3-47
replace contents, (V1)6-144, (V1)7-147
restore from tape, (V2)3-48, (V2)3-53
rewind, (V1)6-154, (V1)7-149
save, (V2)3-54, (V2)3-57
segment definition, (V1)5-55
sequential access, (V1)4-35
shared, (V1)3-12, (V1)4-40
size allocation, (V1)3-12, (V1)4-30, (V1)4-31, (V1)5-55, (V2)3-14, (V2)3-15, (V2)3-65
size extension, (V1)4-30, (V1)6-70, (V1)7-76
sort contents, (V2)8-1
space, (V1)4-30, (V1)4-31
space allocation, (V1)6-206, (V1)6-211
system, (V1)1-9
temporary, (V1)1-9, (V1)4-29, (V1)4-41, (V1)5-14
temporary to permanent, (V1)4-42, (V1)6-175, (V1)6-248, (V1)7-177
truncate, (V1)6-177, (V1)7-179
types, (V1)3-14
update mode, (V1)4-39
write mode, (V1)4-38
zeroing, (V1)3-13
File Assignment Table (FAT), (V1)5-20, (V1)5-51
File Control Block, see FCB
File Management
disk file protection, (V1)1-9
multiprocessor files, (V1)1-9
permanent files, (V1)1-8
random access files, (V1)1-9
system files, (V1)1-9
temporary files, (V1)1-9
File Overlap Detection Utility, (V4)2-27

File Pointer Table (FPT), L-46
Firmware
  display, (V3)12-5
  load (ACS/WCS), (V3)12-4
  load (DEVINITL), (V3)11-3
Fixed Head Disk, see FHD
Flag
  reset (false), (V2)1-88
  set (true), (V2)1-95
Floppy Disk
  archive, (V4)2-42
  duplication, (V4)2-42
  format, (V4)2-42
  media initialization (J.FORMF), (V3)10-42
Format
  floppy disk, (V3)10-42
  memory disk, (V3)10-59
FPT, L-46
Free Dynamic Extended Indexed Data Space Service, (V1)6-72
Free Dynamic Task Execution Space Service, (V1)6-73
Free Memory in Byte Increments Service, (V1)6-111, (V1)7-79
Free Shared Memory Service, (V1)6-233
FSORT2
  access, (V2)8-8
  directives
    field, (V2)8-14
    header, (V2)8-13
    usage notes, (V2)8-15
  examples, (V2)8-12, (V2)8-17
  extra memory, (V2)8-7
  format
    blocked, (V2)8-5
    direct access blocked, (V2)8-6
    direct access unblocked, (V2)8-6
    special blocked, (V2)8-6
    special unblocked, (V2)8-6
    unblocked, (V2)8-6
  input data elements, (V2)8-5
  introduction, (V2)8-1
  logical file codes, (V2)8-1, (V2)8-5
  options, (V2)8-7
  subroutines
    SORT:FLD, (V2)8-11
    SORT:HDR, (V2)8-10
    SORT:PAR, (V2)8-9
    SORT:X, (V2)8-12

- G -

Get Address Limits Service, (V1)6-76, (V1)6-77
Get Base Mode Task Address Limits Service, (V1)7-103
Get Command Line Service, (V1)6-24, (V1)7-30
Get Current Date and Time Service, (V1)7-84
Get Definition for Terminal Function Service, (V1)6-81, (V1)7-81
Get Device Mnemonic or Type Code Service, (V1)6-52, (V1)7-60
Get Dynamic Extended Data Space Service, (V1)6-78
Get Dynamic Extended Discontiguous Data Space Service, (V1)6-79
Get Dynamic Task Execution Space Service, (V1)6-80
Get Extended Memory Array Service, (V1)6-217, (V1)7-211
Get Memory in Byte Increments Service, (V1)6-110, (V1)7-83
Get Message Parameters Service, (V1)6-83, (V1)7-86
Get Real Physical Address Service, (V1)6-139, (V1)7-140
Get Run Parameters Service, (V1)6-84, (V1)7-87
Get Shared Memory Service, (V1)6-242
Get Task Environment Service, (V1)6-66, (V1)7-72
Get Task Number Service, (V1)6-88, (V1)6-118, (V1)7-91, (V1)7-120
Get TSA Start Address Service, (V1)6-86, (V1)7-89
Get User Context Service, (V1)7-80
Global Common, (V1)1-11, (V1)3-19, (V1)3-20, (V3)7-39
GPMC Devices, specification, (V1)5-28
Granularity, (V1)4-30

- H -

H.ALOC, (V1)5-50
H.BKDM, (V1)5-34
H.DBUG1, see System Debugger
H.DBUG2, see System Debugger
H.ICP, (V4)2-32
H.MDT, (V3)7-35
H.MDXIO, (V3)10-58
H.MONS, (V1)5-49
H.MVMT, (V1)5-22
H.REMM, (V1)5-1, (V1)5-4, (V1)5-50
  status codes, (V1)5-17
H.REXS, (V1)5-49
H.VOMM, (V1)5-20, (V1)5-51
  status codes, (V1)5-57
  system services, (V1)5-65
Handlers, device, (V3)7-15
Hardware
  channel configuration, (V3)7-13

device characteristics, (V3)10-19
disk descriptions, (V1)5-27
I/O device definitions, (V3)7-18
machine type configuration, (V3)7-34
memory disk configuration, (V3)7-13
minimum configuration, (V1)1-22
shadow memory configuration, (V3)10-56
starter system addresses, (V3)2-1
Hardware Interrupts/Traps, (V1)1-5
Hardware Priorities, (V1)1-5
HELP, (V2)9-1
Help, online, see Online Help
HELP, VOLMGR, (V2)3-41
Help Files
description, (V2)9-9
offline, (V2)2-35
online, (V2)2-36
specify location, (V3)7-30
HELPT, (V2)9-1, (V2)9-14
HSD Interface, FCB settings, L-39

- I -

Identify a Job, (V2)1-72
Implicit Mount Management, (V3)7-13
Include Memory Partition Service, (V1)6-90
Include Shared Image Service, (V1)7-93
Increase File Size, (V2)3-41
Increase Memory Allocation, (V2)1-35
Indicate Object Records, (V2)1-80
Indirectly Connected Interrupt Response Timing Module, (V4)2-32
Indirectly Connected Task Linkage Block (ITLB), (V3)7-32
Inhibit
banner page, (V2)2-31, (V3)7-33, (V3)7-36
demand page, (V3)7-40
mount message, (V2)1-77, (V2)2-31, (V2)2-33, (V3)7-36
operator intervention, (V2)2-31, (V3)7-36, (V3)10-44
public volume dismount, (V3)7-13
INIT
description, (V3)10-19
errors, (V3)10-39
LOGONFLE, see LOGONFLE
Initial Program Load (IPL), (V3)6-1
Initialize
ACM, (V3)10-19
ACM/MFP, (V3)10-35
ALIM, (V3)10-19, (V3)10-26
devices (DEVINITL), (V3)11-1
disks online, (V3)13-20
DP II, (V3)13-12, (V3)13-21
floppy disk, (V3)10-42
non-TSM devices, (V3)10-20
SCSI disk, (V3)13-12, (V3)13-23
TSM devices, (V3)10-20
UDP, (V3)13-12, (V3)13-21
Input/Output, see I/O
Input/Output Control System (IOCS), (V1)1-10, (V1)5-1, (V1)5-38
Install
a starter system, (V3)2-1
a user SDT, (V3)4-3
INSTALLSDT, (V3)2-17, (V3)2-18
Instruction Sequence Timing Tool, (V4)2-38
Integers, powers of, E-1
Interactive Environment
accessing, (V2)1-12
sample task, (V2)1-107
priority level, (V3)7-62
Interactive String Search, (V4)2-52
Interactive Task, (V2)1-25
Internal Processing Unit, see IPU
Interrupt Processors, (V1)2-51, (V3)7-46
Interrupt, task, see Task, interrupt
Intertask Communication, (V1)1-11, (V1)2-22, (V1)2-23, (V1)6-162, (V1)7-166
run requests, (V1)1-11
user status word, (V1)6-155, (V1)6-183, (V1)7-159, (V1)7-185
I/O
blocked, (V1)5-34
device definitions, (V3)7-18
device-dependent, (V1)5-43
device-independent, (V1)1-10, (V1)5-1, (V1)5-32
direct, (V1)1-10, (V1)5-34
direct channel, (V1)5-43
file access, (V1)1-11
IOCS, (V1)1-10
logical, (V1)5-1
logical file codes, see LFC
no-wait, (V1)5-33
processing, (V1)1-10, (V1)5-32, (V1)5-38
scheduling, (V1)2-42
status, (V1)5-38
terminate in process, (V2)1-19, (V2)1-27
wait, (V1)5-32
IOCS, (V1)1-10, (V1)5-1, (V1)5-38
IOQ Memory Pool, specify size, (V3)7-31
IPL, automatic, (V3)6-4
IPU
accounting, (V1)2-17
configuration, (V3)7-32
CPU execution of tasks, (V1)2-16
display status, (V2)2-46
display traps, (V2)2-26
executable system services, (V1)2-18
base, (V1)7-5

nonbase, (V1)6-3
general description, (V1)2-15
offline, (V2)2-35
online, (V2)2-36
options, (V1)2-15
priority versus biasing, (V1)2-17
scheduling, (V1)2-18
set bias, (V1)6-98, (V1)7-102
task prioritization
biased, (V1)2-15
nonbiased, (V1)2-16
task selection, (V1)2-16
IPU/CPU Scheduler, (V1)2-16
selection, (V3)7-17

- J -

J.DSCMP
description, (V3)14-1
disk status report, (V3)14-3
error messages, (V3)14-6
logical file codes, (V3)14-2
performance, (V3)14-2
segment report, (V3)14-3, (V3)14-4
usage, (V3)14-3
J.DTSAVE, (V3)7-27, (V3)10-44
J.FORMF, (V3)10-42
J.HLP, (V2)9-1, (V3)7-30
J.INIT
conventions, (V3)9-2
dedicated names, (V3)9-2
directive summary, (V3)9-3
directives
Change Contents of Memory Location, (V3)9-3
Comments, (V3)9-8
Conditional, (V3)9-6
Define Base Address, (V3)9-3
Define Named Value, (V3)9-4
Define Patch Area, (V3)9-6
Enter Value into Patch Area, (V3)9-7
Exit, (V3)9-4
Go to Patch Area, (V3)9-5
Return from Patch Area, (V3)9-6
Select Patch Options, (V3)9-5
entry conditions, (V3)9-8
examples, (V3)9-10
exit conditions, (V3)9-8
external references, (V3)9-9
introduction, (V3)9-1
J.LABEL, (V3)10-62
J.MDREST, (V3)10-58, (V3)10-61
J.MDSAVE, (V3)10-58, (V3)10-60
J.MDTI
access, (V2)5-2
contents, (V2)5-1
description, (V2)5-1
errors, (V2)5-8
examples, (V2)5-6
exit, (V2)5-4
input files, (V2)5-4
logical file codes, (V2)5-3, (V2)5-4
programming considerations, (V2)5-7
J.MOUNT, (V1)1-13
J.SHAD
accessing, (V2)6-1
directives
EXIT, (V2)6-3
SHADOW, (V2)6-3
errors, (V2)6-4
examples, (V2)6-5
introduction, (V2)6-1
logical file codes, (V2)6-2
J.SHUTD
error messages, (V3)10-48
using, (V3)10-45
J.TDEFI Program, (V2)11-4
J.TSET Utility, (V2)11-21
J.UNLOCK, (V2)2-58, (V3)10-53
J.VFMT, (V1)1-17, (V3)13-1
access, (V3)13-4
CONFIRM option, (V3)13-4
directive syntax, (V3)13-3
directives
COPY, (V3)13-5
EDITFMAP, (V3)13-7
EXIT, (V3)13-9
FORMAT, (V3)13-9
INITIALIZE, (V3)13-12
NEWBOOT, (V3)13-16
REPLACE, (V3)13-17
errors, (V3)13-18
examples, (V3)13-18
introduction, (V3)13-1
logical file codes, (V3)13-1
media management, see Media Management
usage, (V3)13-2
JCL
conditional processing, (V2)1-29
directive files, (V2)1-27
directive summary, (V2)1-2
directive syntax, (V2)1-33
directives, see TSM, directives
introduction, (V2)1-1
macro looping, (V2)1-29
parameter passing, (V2)1-29
parameter replacement, (V2)1-30
spooled input control, (V2)1-30
Job Accounting File, display, (V2)1-33, (V2)2-26
Job Accounting Program (M.ACCNT), (V3)10-15

Job Activation, (V1)6-226
also see Task, execution
Job Control Language, see JCL
Job Identification, (V2)1-72
Job Number, (V2)2-5
Job Processing
data flow, (V2)1-31
terminating conditions, (V2)1-32
Job Queue, display, (V2)1-99

- K -

Key, (V2)1-9, (V3)10-4
KEY, (V1)1-20, (V3)10-2
directive summary, (V3)10-4
directives
ADD, (V3)10-4
CHANGE, (V3)10-7
DEFAULTS, (V3)10-8
DELETE, (V3)10-9
LOG, (V3)10-9
NEWFILE, (V3)10-9
X, (V3)10-10
examples, (V3)10-10
M.KEY file, (V3)10-3
usage, (V3)10-3
KEYWORD Task, (V2)1-9
Kill a Task, (V2)2-25

- L -

Label, establish, (V2)1-52
Label ANSI Tape Utility (J.LABEL), (V3)10-62
Large Buffers, (V1)3-15
Laser Printer Utility, (V4)2-34
LFC
assignments, (V2)1-21, (V2)1-36
in FCB, (V1)5-38
overview, (V1)5-2
LIBED, (V1)1-17
Libraries
scientific subroutine, (V1)1-21
subroutine, (V1)1-21
system macro, (V1)1-21
user group, (V1)1-21
Linker/X32, (V1)1-18
LINKX32, (V1)1-18
List, see Display
LMIT, (V1)2-7
Load and Execute Interactive Debugger Service, (V1)6-45, (V1)7-55
Load Module Information, (V4)2-35
Load Module Information Table (LMIT), (V1)2-7
Load Overlay Segment Service, (V1)6-121
Load Shared Image into Memory, (V2)2-24
LOADACS, see ACS
Log Contents of Rapid File Allocation MDT, (V4)2-35
Log Off the System, (V2)1-14, (V2)1-59
Log On the System, (V2)1-8
multiple logons, (V2)1-8, (V3)7-33
Log Resource or Directory Service, (V1)6-103, (V1)7-106
Logged on Users, display, (V2)1-106
Logical Address
task boundaries, (V1)6-76, (V1)6-77
verify, (V1)6-194, (V1)7-196
Logical Channel Program, (V1)5-43, (V1)5-44
Logical Dismount, (V1)4-18
Logical File Codes, see LFC
Logical I/O, (V1)5-1
Logical Mount, (V1)4-18
Logoff, remote terminal, (V4)2-37
Logon
disable, (V2)1-54
enable, (V2)1-57
multiple, (V2)1-8, (V3)7-33
SYSTEM, (V3)7-33, (V3)7-61
Logon Attempt Counter, (V3)10-22
Logon Timeout Counter, (V3)10-23
LOGONFLE, (V2)11-17, (V3)10-20
format, (V3)10-20
sample, (V3)10-38
security counters, (V3)10-22
security examples, (V3)10-24

- M -

M.ACCNT File, (V3)10-11, (V3)10-15
delete contents, (V2)2-37
M.ACS, see ACS
M.ACTV, (V1)2-7, (V1)6-5
M.ADRS, (V1)6-6
M.ALOC, (V1)6-222
M.ANYW, (V1)2-25, (V1)6-7
M.ASSN, (V1)6-8
M.ASYNCH, (V1)6-10
M.BACK, (V1)6-11
M.BATCH, (V1)6-13, (V4)2-17
M.BBTIM, (V1)6-15
M.BORT, (V1)2-46, (V1)6-16
M.BRK, (V1)2-21, (V1)6-19
M.BRKXIT, (V1)2-21, (V1)6-19
M.BTIM, (V1)6-20
M.CDJS, (V1)6-226
M.CLOSER, (V1)6-21
M.CLSE, (V1)6-23
M.CMD, (V1)6-24
M.CNTRL File, (V3)10-17
M.CONABB, (V1)6-25
M.CONADB, (V1)6-26
M.CONAHB, (V1)6-27

M.CONASB, (V1)6-28
M.CONBAD, (V1)6-29
M.CONBAF, (V1)6-30
M.CONBAH, (V1)6-31
M.CONBBA, (V1)6-32
M.CONBBY, (V1)6-33
M.CONBYB, (V1)6-34
M.CONN, (V1)6-35
M.CPERM, (V1)6-37
M.CREATE, (V1)6-228
M.CTIM, (V1)6-39
M.CWAT, (V1)6-41
M.DALC, (V1)6-231
M.DASN, (V1)6-42
M.DATE, (V1)6-44
M.DEBUG, (V1)6-45
M.DEFT, (V1)6-46
M.DELETE, (V1)6-232
M.DELR, (V1)6-48
M.DELTSK, (V1)2-46, (V1)6-50
M.DEVID, (V1)6-52
M.DIR, (V1)6-53
M.DISCON, (V1)6-55
M.DLTT, (V1)6-56
M.DMOUNT, (V1)6-57
M.DSMI, (V1)6-59
M.DSUB, (V1)6-60
M.DUMP, (V1)6-61
M.EAWAIT, (V1)2-25, (V1)6-63
M.ENMI, (V1)6-64
M.ENUB, (V1)6-65
M.ENVRMT, (V1)6-66
M.ERR File, (V3)10-18
M.EXCL, (V1)6-233
M.EXCLUDE, (V1)3-20, (V1)6-67
M.EXIT, (V1)2-24, (V1)2-46, (V1)6-69
M.EXTD, (V1)6-70
M.FADD, (V1)6-234
M.FD, (V1)2-43, (V1)3-19, (V1)6-72
M.FE, (V1)2-43, (V1)3-19, (V1)6-73
M.FILE, (V1)6-236
M.FSLR, (V1)6-237
M.FSLS, (V1)6-238
M.FWRD, (V1)6-74
M.FXLR, (V1)6-240
M.FXLS, (V1)6-241
M.GADRL, (V1)6-76
M.GADRL2, (V1)6-77
M.GD, (V1)2-43, (V1)3-19, (V1)6-78,
M.GDD, (V1)6-79
M.GE, (V1)2-43, (V1)3-19, (V1)6-80
M.GETDEF, (V1)6-81, (V2)11-18
  errors, (V2)11-19
M.GMSGP, (V1)2-22, (V1)2-24, (V1)6-83
M.GRUNP, (V1)2-22, (V1)2-24, (V1)6-84
M.GTIM, (V1)6-85
M.GTSAD, (V1)6-86
M.HOLD, (V1)6-87
M.ID, (V1)6-88
M.INCL, (V1)6-242
M.INCLUDE, (V1)3-20, (V1)6-90
M.INQUIRY, (V1)6-93
M.INT, (V1)6-97
M.IPUBS, (V1)6-98
M.KEY Editor, see KEY
M.KEY File, (V1)4-13, (V3)10-3
M.LOC, (V1)6-99
M.LOCK, (V1)6-101
M.LOG, (V1)6-244
M.LOGR, (V1)6-103
M.MACLIB, (V1)1-21
M.MEM, (V1)6-108
M.MEMB, (V1)2-43, (V1)3-19, (V1)6-110
M.MEMFRE, (V1)2-43, (V1)3-19, (V1)6-111
M.MOD, (V1)6-112
M.MODU, (V1)6-114
M.MOUNT, (V1)6-115
M.MOUNT File, (V3)10-43
M.MOVE, (V1)6-117
M.MPXMAC, (V1)1-21
M.MYID, (V1)6-118
M.NEWRRS, (V1)6-119
M.OLAY, (V1)6-121
M.OPENR, (V1)6-122
M.OSREAD, (V1)6-124
M.OSWRIT, (V1)6-125
M.PDEV, (V1)6-246
M.PERM, (V1)6-248
M.PGOD, (V1)6-126
M.PGOW, (V1)6-127
M.PNAM, (V1)6-128
M.PNAMB, (V1)6-129
M.PRIL, (V1)6-131
M.PRIV, (V1)6-132
M.PRJCT File, (V3)10-11
M.PTSK, (V1)2-7, (V1)6-133
M.QATIM, (V1)6-138
M.RADDR, (V1)6-139
M.RCVR, (V1)2-22, (V1)2-23, (V1)6-140
M.READ, (V1)6-141
M.RELP, (V1)6-142
M.RENAM, (V1)6-143
M.REPLAC, (V1)6-144
M.RESP, (V1)6-145
M.REWRIT, (V1)6-146
M.REWRTU, (V1)6-147
M.ROPL, (V1)6-148
M.RRES, (V1)6-149
M.RSML, (V1)6-150
M.RSRV, (V1)6-153
M.RWND, (V1)6-154
M.SETS, (V1)6-155

M.SETSYNC, (V1)6-157
M.SETT, (V1)6-159
M.SHARE, (V1)6-250
M.SMSGR, (V1)2-25, (V1)6-162
M.SMULK, (V1)6-252
M.SOPL, (V1)6-163
M.SRUNR, (V1)2-26, (V1)6-164
M.SUAR, (V1)2-34, (V1)6-166
M.SUME, (V1)6-167
M.SURE, (V1)6-168
  context switch timing, (V4)2-20
M.SUSP, (V1)2-25, (V1)6-169
M.SYNCH, (V1)6-170
M.TBRKON, (V1)6-171
M.TDAY, (V1)6-172
M.TEMP, (V1)6-173
M.TEMPER, (V1)6-175
M.TRNC, (V1)6-177
M.TSCAN, (V1)6-178
M.TSMPC, (V1)6-179
M.TSTE, (V1)6-182
M.TSTS, (V1)6-183
M.TSTT, (V1)6-184
M.TURNON, (V1)6-185
M.TYPE, (V1)6-187
M.UNLOCK, (V1)6-188
M.UNSYNC, (V1)6-190
M.UPRIV, (V1)6-192
M.UPSP, (V1)6-193
M.USER, (V1)6-253
M.VADDR, (V1)6-194
M.WAIT, (V1)6-195
M.WEOF, (V1)6-196
M.WRIT, (V1)6-197
M.XBRKR, (V1)6-198
M.XIEA, (V1)2-22, (V1)6-199
M.XMEA, (V1)2-22, (V1)2-26, (V1)6-200
M.XMSGR, (V1)2-22, (V1)2-24, (V1)6-201
M.XREA, (V1)2-22, (V1)2-26, (V1)6-202
M.XRUNR, (V1)2-22, (V1)2-24, (V1)6-203
M.XTIME, (V1)6-204
M_ACTV, (V1)7-7
M_ADRS, (V1)7-8
M_ADVANCE, (V1)7-9
M_ANYWAIT, (V1)7-11
M_ASSIGN, (V1)7-12
M_ASYNCH, (V1)7-14
M_AWAITACTION, (V1)7-15
M_BACKSPACE, (V1)7-16
M_BATCH, (V1)7-18
M_BBTIM, (V1)7-20
M_BORT, (V1)7-21
M_BRK, (V1)7-24
M_BRKXIT, (V1)7-24
M_BTIM, (V1)7-25
M_CHANPROGFCB, (V1)7-26
M_CLOSER, (V1)7-27
M_CLSE, (V1)7-29
M_CMD, (V1)7-30
M_CONABB, (V1)7-31
M_CONADB, (V1)7-32
M_CONAHB, (V1)7-33
M_CONASB, (V1)7-34
M_CONBAD, (V1)7-35
M_CONBAF, (V1)7-36
M_CONBAH, (V1)7-37
M_CONBBA, (V1)7-38
M_CONBBY, (V1)7-39
M_CONBYB, (V1)7-40
M_CONN, (V1)7-41
M_CONSTRUCTPATH, (V1)7-42
M_CONVERTTIME, (V1)7-43
M_CREATEFCB, (V1)7-45
M_CREATEP, (V1)7-46
M_CREATET, (V1)7-48
M_CTIM, (V1)7-50
M_CWAT, (V1)7-51
M_DATE, (V1)7-52
M_DEASSIGN, (V1)7-53
M_DEBUG, (V1)7-55
M_DEFT, (V1)7-56
M_DELETER, (V1)7-57
M_DELTSK, (V1)7-59
M_DEVID, (V1)7-60
M_DIR, (V1)7-61
M_DISCON, (V1)7-63
M_DISMOUNT, (V1)7-64
M_DLTT, (V1)7-66
M_DSMI, (V1)7-67
M_DSUB, (V1)7-68
M_DUMP, (V1)7-69
M_ENMI, (V1)7-70
M_ENUB, (V1)7-71
M_ENVRMT, (V1)7-72
M_EXCLUDE, (V1)7-73
M_EXIT, (V1)2-46, (V1)7-75
M_EXTENDFILE, (V1)7-76
M_EXTSTS, (V1)7-78
M_FREEMEMBYTES, (V1)7-79
M_GETCTX, (V1)7-80
M_GETDEF, (V1)7-81
M_GETMEMBYTES, (V1)7-83
M_GETTIME, (V1)7-84
M_GMSGP, (V1)7-86
M_GRUNP, (V1)7-87
M_GTIM, (V1)7-88
M_GTSAD, (V1)7-89
M_HOLD, (V1)7-90
M_ID, (V1)7-91
M_INCLUDE, (V1)7-93
M_INQUIRER, (V1)7-96
M_INT, (V1)7-101

M_IPUBS, (V1)7-102
M_LIMITS, (V1)7-103
M_LOCK, (V1)7-104
M_LOGR, (V1)7-106
M_MEM, (V1)7-111
M_MOD, (V1)7-113
M_MODU, (V1)7-115
M_MOUNT, (V1)7-116
M_MOVE, (V1)7-118
M_MYID, (V1)7-120
M_OPENR, (V1)7-121
M_OPTIONDWORD, (V1)7-124
M_OPTIONWORD, (V1)7-125
M_OSREAD, (V1)7-126
M_OSWRIT, (V1)7-127
M_PNAMB, (V1)7-129
M_PRIL, (V1)7-131
M_PRIVMODE, (V1)7-132
M_PTSK, (V1)7-133
M_PUTCTX, (V1)7-138
M_QATIM, (V1)7-139
M_RADDR, (V1)7-140
M_RCVR, (V1)7-141
M_READ, (V1)7-142
M_READD, (V1)7-144
M_RELP, (V1)7-145
M_RENAME, (V1)7-146
M_REPLACE, (V1)7-147
M_RESP, (V1)7-148
M_REWIND, (V1)7-149
M_REWRIT, (V1)7-150
M_REWRTU, (V1)7-151
M_ROPL, (V1)7-152
M_RRES, (V1)7-153
M_RSML, (V1)7-154
M_RSMU, (V1)7-155
M_RSRV, (V1)7-156
M_SETERA, (V1)2-37, (V1)7-157
M_SETEXA, (V1)2-37, (V1)7-158
M_SETS, (V1)7-159
M_SETSYNC, (V1)7-161
M_SETT, (V1)7-163
M_SMSGR, (V1)7-166
M_SOPL, (V1)7-167
M_SRUNR, (V1)7-168
M_SUAR, (V1)7-170
M_SUME, (V1)7-171
M_SURE, (V1)7-172
M_SUSP, (V1)7-173
M_SYNCH, (V1)7-174
M_TBRKON, (V1)7-175
M_TDAY, (V1)7-176
M_TEMPFILETOPERM, (V1)7-177
M_TRUNCATE, (V1)7-179
M_TSCAN, (V1)7-180
M_TSMPC, (V1)7-181
M_TSTE, (V1)7-184
M_TSTS, (V1)7-185
M_TSTT, (V1)7-186
M_TURNON, (V1)7-187
M_TYPE, (V1)7-189
M_UNLOCK, (V1)7-190
M_UNPRIVMODE, (V1)7-192
M_UNSYNC, (V1)7-193
M_UPSP, (V1)7-195
M_VADDR, (V1)7-196
M_WAIT, (V1)7-197
M_WRITE, (V1)7-198
M_WRITEEOF, (V1)7-199
M_XBRKR, (V1)7-200
M_XIEA, (V1)7-201
M_XMEA, (V1)7-202
M_XMSGR, (V1)7-203
M_XREA, (V1)7-204
M_XRUNR, (V1)7-205
M_XTIME, (V1)7-206
Machine Type, (V3)2-27, (V3)7-34
MACLIBR, (V1)1-16
Macro Assembler, (V1)1-16
Macro Directive Files, (V2)1-30
Macro Librarian/X32, (V1)1-17
Macro Libraries, (V1)1-21
Macro Library Editor, (V1)1-16
Macro Looping, (V2)1-29
MACX32, (V1)1-17
Magnetic Tape
  advance, (V1)6-74, (V1)7-9, (V2)3-64, (V2)3-65
  backspace, (V2)3-22
  convert, (V2)3-23
  multivolume
    description, (V1)5-22
    write EOF, (V1)6-193, (V1)6-196, (V1)6-197, (V1)6-214, (V1)6-215, (V1)7-195, (V1)7-198, (V1)7-199, (V1)7-208, (V1)7-209
    write volume record, (V1)6-193, (V1)6-196, (V1)6-197, (V1)6-214, (V1)6-215, (V1)7-195, (V1)7-198, (V1)7-199, (V1)7-208, (V1)7-209
  restore files, (V2)3-48, (V2)3-53
  rewind, (V2)3-53
Map Block, (V1)3-16
  address assignments, J-1
Mapped Out, (V1)3-37
Mapped Programming Executive, see MPX-32
Master System Distribution Tape, see SDT
MDT, (V3)7-35
MEDIA, (V1)1-19
Media, unformatted, (V1)5-25
Media Conversion, (V1)1-19
Media Flaw Data, (V3)13-21

Media Flaw Map, (V3)13-23
Media Management
during SDT boot, (V3)13-22
edit media flaw data, (V3)13-21
EDITFMAP directive, (V3)13-7
INITIALIZE directive, (V3)13-12
initialize media, (V3)13-20
media flaw map, (V3)13-23
SCSI disks, (V3)13-23
terminology, (V3)13-19
Memory
deallocate, (V1)6-111, (V1)7-79
display, (V2)2-46
physical read, (V1)6-124, (V1)7-126
physical write, (V1)6-125, (V1)7-127
search, (V2)2-41
shadow, (V2)1-98, (V3)10-54
Memory Address
display, (V2)2-41
get physical, (V1)6-139, (V1)7-140
inquiry, (V1)6-6, (V1)7-8
Memory Address Inquiry Service, (V1)6-6, (V1)7-8
Memory Allocation
blocked I/O, (V1)3-15
deallocate map block, (V1)6-72, (V1)6-73
demand page, (V1)3-17
dynamic, (V1)1-8, (V1)3-19
expand, (V1)6-80, (V1)6-110, (V1)7-83
extended area, (V1)6-78, (V1)6-79
extended data space, (V1)3-20
increase, (V2)1-35
map block, (V1)3-16
MPX-32, (V1)3-15
static, (V1)3-18
task, (V1)3-16
Memory Classes, (V1)3-16, (V3)10-56
Memory Disk
abort cases, (V3)10-59
access, (V3)10-59
configuration, (V3)7-13, (V3)10-59
dismount, (V3)10-59
errors, (V3)10-60
format, (V3)10-59
mount, (V3)10-59
overview, (V3)10-58
restore task (J.MDREST), (V3)10-61
save task (J.MDSAVE), (V3)10-60
usage, (V3)10-59
Memory Dump Request Service, (V1)6-61, (V1)7-69
Memory Partition, (V1)3-20
access, (V1)4-44
access attributes, (V1)3-10
attach, (V1)4-44
create, (V1)4-43, (V1)6-108, (V1)6-250, (V1)7-111, (V2)3-29
Datapool, see Datapool
define static, (V3)7-39
delete, (V1)4-44, (V1)6-232, (V2)3-37
detach, (V1)4-44
display, (V2)3-44
dynamic, (V1)3-18
exclude, (V1)6-67
extended common, (V1)3-19
global common, see Global Common
include, (V1)6-90, (V1)6-242
nonbase addressing, (V1)4-43
protection, (V1)3-10, (V1)3-20, (V1)4-43
RCB, (V1)5-63, L-64
share, (V1)4-45
static, (V1)3-18
unlock, (V1)6-252
Memory Pool, (V2)1-115
IOQ size, (V3)7-31
MSG size, (V3)7-38
size, (V3)7-46
system, (V1)3-15
Memory Pool Monitor, (V4)2-42
Memory Resident Descriptor Table (MDT), (V2)5-1, (V2)5-7, (V3)7-35
Memory Size, (V3)7-51
Memory Types, (V1)3-16, (V3)10-56
Memory Word, reset, (V2)2-32
Message
end-action processing, (V1)2-26
inhibit batch, (V3)7-11
maximum no-wait, (V3)7-36
receive from other tasks, (V1)2-23, (V1)6-140, (V1)7-141
send to console, (V1)6-187, (V1)7-189, (V3)7-11
send to task, (V1)2-25, (V1)6-162, (V1)7-166, (V2)2-42
send to terminal, (V2)1-80, (V2)1-101, (V3)7-11
system files, (V3)10-18
Message End-Action Routine Exit, (V1)6-200, (V1)7-202
Message Parameters, (V1)2-24, (V1)6-83, (V1)7-86
Message Receiver
establish, (V1)2-23
exit, (V1)2-24, (V1)6-201, (V1)7-203
Minimum Hardware Configuration, (V1)1-22
Modify
page size, (V2)1-83, (V3)7-21
screen width, (V2)1-75, (V3)7-21
Modify Descriptor Service, (V1)6-112, (V1)7-113

Modify Descriptor User Area
Service, (V1)6-114, (V1)7-115
Modify Swap Parameters, (V4)2-53
Mount
memory disk, (V3)10-59
multiprocessor volume, (V1)4-49
public volume, (V3)9-11
volume, (V1)4-49, (V1)6-115, (V1)7-116, (V2)1-77, (V2)2-33
volume (M.MOUNT), (V3)10-43
Mount Volume Service, (V1)6-115, (V1)7-116
Move Data to User Address Service, (V1)6-117, (V1)7-118
Move Non-base TSA, (V1)3-28, (V1)3-32, (V2)1-60
MPX-32
batch processing, (V1)1-14
build, (V3)1-1, (V3)3-1
command processors, (V1)1-3
communications facilities, (V1)1-11
CPU scheduling, (V1)1-7
data transfers between revisions, (V1)5-22
delimiters, (V2)1-26
deliverable software, (V3)2-3
demonstration package, (V4)1-4
extended, see Extended MPX-32
features, (V1)1-3
file management, (V1)1-8
input/output operations, (V1)1-10
installation, (V3)2-1
introduction, (V1)1-1
maintenance, (V3)1-1
mapped in, (V2)1-79
mapped out, (V1)3-37, (V2)1-76
memory allocation, (V1)1-8, (V1)3-15
priority levels, (V1)1-7
recovery, (V3)6-1
restart, (V3)5-1
shutdown, (V3)10-45
software interrupt system, (V1)1-7
system administration, (V1)4-13, (V3)10-1
system description, (V1)1-1
system services, (V1)1-10, (V1)6-1, (V1)7-1
test, (V3)3-1
time management, (V1)1-12
timer scheduler, (V1)1-12
trap processors, (V1)1-12
utilities, see Utilities
MPX.PRO
customizing, (V2)10-7
description, (V2)10-6
errors, (V2)10-9
predefined functions, (V2)10-6
sample file, (V2)10-10
TSM special keys, (V2)1-15
MPXDEMO, (V4)1-4
MSG Memory Pool, specify size, (V3)7-38
Multicopied Tasks, (V1)2-3
Multiprocessor, recovery, (V2)2-58, (V3)10-53
Multiprocessor Lock, (V1)4-47
Multiprocessor Recovery Task (J.UNLOCK), (V3)10-53
Multiprocessor Resource, (V1)4-47, (V1)4-51, (V3)7-27
Multiprocessor Shared Memory, (V1)3-22, (V3)7-52, (V3)7-55
Multiprocessor Shared Volume, (V1)4-47, (V1)4-49
Multiprocessor User Volume, (V1)4-16
MVT, (V1)5-51

- **N** -

No-Wait I/O, (V1)5-33
No-Wait I/O End-Action Return Service, (V1)6-199, (V1)7-201
No-Wait I/O Requests, maximum number, (V3)7-36
No-Wait Messages, maximum number, (V3)7-36
No-Wait Run Requests, maximum number, (V3)7-38
Nonpublic Volume, (V1)4-16
NULL Device, specification, (V1)5-28, (V3)7-25
Numerical Information, D-1

- **O** -

Object Librarian/X32, (V1)1-18
Object Records, indicate, (V2)1-80
OBJX32, (V1)1-18
Offline
device, (V2)2-35
help files, (V2)2-35
IPU, (V2)2-35
Online
device, (V2)2-36
help files, (V2)2-36
IPU, (V2)2-36
Online Disk Media Management, (V3)13-19
also see Media Management
Online Help
access, (V2)9-3
choices within, (V2)9-5
components, (V2)9-1
description, (V2)9-1
display sample, (V2)9-2
errors, (V2)9-15
help files
description, (V2)9-9
offline, (V2)2-35
online, (V2)2-36

specify location, (V3)7-24
translate, (V2)9-14
help key, (V2)9-3
HELPT, (V2)9-1, (V2)9-14
J.HLP, (V3)7-30
keywords, (V2)9-2, (V2)9-12
modify information, (V2)9-8
print help screen, (V2)9-6
sample
display, (V2)9-2
topic entry, (V2)9-11
topic entries, (V2)9-2, (V2)9-10
translate files, (V2)9-14
Online Restart, (V3)5-1
Online System Patch Facility, see J.INIT
OPCOM, (V1)1-14, (V2)2-1
activate, (V2)2-3
batch jobs, (V2)2-5
directive abort, (V2)2-6
directive summary, (V2)2-1
directive syntax, (V2)2-5
directives
ABORT, (V2)2-6
ACTIVATE, (V2)2-7
BATCH, (V2)2-8
BREAK, (V2)2-9
CONNECT, (V2)2-10
CONTINUE, (V2)2-11
DEBUG, (V2)2-12
DELETETIMER, (V2)2-12
DEPRINT, (V2)2-13
DEPUNCH, (V2)2-14
DISABLE, (V2)2-15
DISCONNECT, (V2)2-15
DISMOUNT, (V2)2-16
DUMP, (V2)2-18
ENABLE, (V2)2-19
ENTER, (V2)2-20
ESTABLISH, (V2)2-21
EXCLUDE, (V2)2-22
EXIT, (V2)2-22
HOLD, (V2)2-23
INCLUDE, (V2)2-24
KILL, (V2)2-25
LIST, (V2)2-26
MODE, (V2)2-31
MODIFY, (V2)2-32
MOUNT, (V2)2-33
OFFLINE, (V2)2-35
ONLINE, (V2)2-36
PURGEAC, (V2)2-37
REDIRECT, (V2)2-37
REPRINT, (V2)2-38
REPUNCH, (V2)2-39
REQUEST, (V2)2-40
RESUME, (V2)2-40
SEARCH, (V2)2-41
SEND, (V2)2-42
SETTIMER, (V2)2-44
SNAP, (V2)2-45
STATUS, (V2)2-46
SYSASSIGN, (V2)2-56
TIME, (V2)2-57
TURNON, (V2)2-58
UNLOCK, (V2)2-58
WAIT, (V2)2-60
exit, (V2)2-22
functionality, (V2)2-1
job numbers, (V2)2-5
owner names, (V2)2-4, (V2)2-5
restrict directives, (V2)2-3
set system operations, (V2)2-31
system console, (V2)2-4
system task restrictions, (V2)2-4
task names, (V2)2-4
task numbers, (V2)2-4
Open, resource, (V1)5-3
Open File Service, (V1)6-236
Open Resource Service, (V1)6-122, (V1)7-121
Operating Environments, (V2)1-11, (V2)1-13
Operator Communications, see OPCOM
Operator Console, see Console
Operator Intervention Inhibit, (V3)10-44
Option Word Inquiry, (V1)6-126, (V1)6-127, (V1)7-124, (V1)7-125
Options
TSM, see TSM, options
VOLMGR, see VOLMGR, options
Others, (V1)3-6, (V1)4-40, (V3)7-42
Overlay, load, (V1)6-121
Owner, (V3)7-43, (V3)10-3
Owner Name, (V2)1-8, (V2)2-4, (V3)10-4
validation, (V1)6-218

- **P** -

Page Size, (V2)1-83, (V3)7-21
Panel Mode Commands, G-1
Parameter, assign a value, (V2)1-94
Parameter Passing, (V2)1-29
Parameter Receive Block (PRB), (V1)2-32, L-54
Parameter Replacement
append a value, (V2)1-30
macro files, (V2)1-30
Parameter Send Block (PSB), (V1)2-27, L-55
Parameter Task Activation Block (PTASK), (V1)6-134, (V1)7-134, L-47
Parameter Task Activation Service, (V1)6-133, (V1)7-133
Password, (V2)1-9, (V3)7-44, (V3)10-4
for terminal ports, (V4)2-43

PASSWORD Task, (V2)1-10, (V3)7-49
Patch Area, (V3)7-45
Patch Facility, see J.INIT
Patch File, (V3)7-45
Pathname,
execution, (V1)4-7
fully qualified, (V1)4-8, (V1)4-10
partially qualified, (V1)4-9, (V1)4-11
reconstruct, (V1)6-128
syntax check, (V1)6-129, (V1)7-129
with VOLMGR, (V2)3-15
Pathname Block (PNB), (V1)5-52, L-50
PCB, (V1)6-180, (V1)7-182, L-50
Peripheral Device
allocate, (V1)6-222
deallocate, (V1)6-231
Permanent File Address Inquiry Service, (V1)6-234
Permanent File Log Service, (V1)6-244
Permanent Files, see File, permanent
Physical Channel Program, (V1)5-43, (V1)5-44
Physical Device Inquiry Service, (V1)6-246
Physical Dismount, (V1)4-19
Physical Memory Read Service, (V1)6-124, (V1)7-126
Physical Memory Write Service, (V1)6-125, (V1)7-127
Physical Mount, (V1)4-17
PNB, (V1)5-52, L-50
Port Protection, (V4)2-43
Post Program-Controlled Interrupt (PPCI),
caller notification packet, (V1)5-47, L-53
end-action receiver, (V1)5-44
Powers of Integers, E-1
PPCI, (V1)5-44, (V1)5-47, L-53
PRB, (V1)2-32, L-54
Print a File, (V2)1-84
Printers
device definition, (V3)7-21, (V3)7-25
laser support, (V4)2-34
serial, formatter/spooler, (V4)2-51
Priority
change batch job, (V2)1-103
change task, (V1)6-131, (V1)7-131
increments, (V1)2-11
levels, (V1)1-7
batch jobs, (V3)7-11
interactive tasks, (V3)7-62
migration, (V1)2-11
task execution, (V1)2-10
Privilege Mode, (V1)6-132, (V1)7-132
Privilege Task, (V1)6-132, (V1)7-132
Process a Different Directive File, (V2)1-46
Program Development Utilities, (V1)1-15
Program Hold Request Service, (V1)6-87, (V1)7-90
PROJECT
directive summary, (V3)10-12
directives
ADD, (V3)10-12
CHANGE, (V3)10-13
DELETE, (V3)10-13
LOG, (V3)10-13
NEWFILE, (V3)10-14
X, (V3)10-14
examples, (V3)10-14
M.PRJCT file, (V3)10-11
usage, (V3)10-11
Project Group, (V2)1-11, (V3)7-48, (V3)10-11
change, (V1)6-46, (V1)7-56, (V2)1-48
user default, (V3)10-4
validate, (V1)6-218
Protection Granule, (V1)3-16
PSB, (V1)2-27, L-55
Pseudonym, (V2)2-27
PTASK Block, (V1)6-134, (V1)7-134, L-47
Public Volume
automatic mount, (V3)9-11
description, (V1)4-16
dismount, (V2)1-55, (V2)2-16
mount, (V2)1-77, (V2)2-33
Purge, (V1)6-214, (V1)7-208
Put User Context Service, (V1)7-138

- R -

Random Access, (V1)1-9, (V1)4-35
Rapid File Allocation Utility, see J.MDTI
RCB
description, (V1)5-59, L-60
directory, (V1)5-62, L-63
file, (V1)5-59, L-60
memory partition, (V1)5-83, L-64
RDTR, (V2)3-6
read from tape, (V2)3-46
Read Descriptor Service, (V1)6-99, (V1)7-144
Read Directives, from a file, (V2)1-89
Read RDTR from Tape, (V2)3-46
Read Record Service, (V1)6-141, (V1)7-142
Read/Lock Write/Unlock (RLWU), (V3)7-48
Read/Write Authorization File Service, (V1)6-218
Real-Time Clock
interrupts per second, (V3)7-39
interrupts per time unit, (V3)7-42
Real-Time Environment, accessing, (V2)1-12
Real-Time Task Accounting, (V1)2-53, (V2)2-31, (V3)7-36
Realtime Debugger, (V4)2-44
Recall Command Lines, (V2)1-85, (V2)10-4
Receive Message Link Address Service, (V1)6-140, (V1)7-141

Receiver Exit Block (RXB), (V1)2-33, L-74
Reconstruct Pathname Service, (V1)6-128, (V1)7-42
Records
backspace, (V1)6-11, (V1)7-16
read, (V1)6-141, (V1)7-142
write, (V1)6-197, (V1)7-198
Recover Multiprocessor, (V2)2-58, (V3)10-53
Recovering the System, (V3)6-1
Redirect SLO/SBO Output, (V2)2-37
Reflective Memory System Software (RMSS), (V1)4-15, (V3)7-41
Reformat RRS Entry Service, (V1)6-119
Reinstate Privilege Mode to Privilege Task Service, (V1)6-132, (V1)7-132
Release Channel Reservation Service, (V1)6-149, (V1)7-153
Release Dual-Ported Disk/Set Dual-Channel ACM Mode Service, (V1)6-142, (V1)7-145
Release Exclusive File Lock Service, (V1)6-240
Release Exclusive Resource Lock Service, (V1)6-188, (V1)7-190
Release FHD Port Service, (V1)6-219, (V1)7-212
Release Synchronization File Lock Service, (V1)6-237
Release Synchronous Resource Lock Service, (V1)6-190, (V1)7-193
Remote Terminal Logoff, (V4)2-37
Remove a Job, (V2)1-86
Remove Shared Image from Memory, (V2)2-22
Rename
file, (V1)6-143, (V1)7-146, (V2)1-87, (V2)3-47
volume, (V4)2-44
Rename File Service, (V1)6-143, (V1)7-146
Replace Permanent File Service, (V1)6-144, (V1)7-147
Reprint SLO, (V2)2-38
Repunch SBO, (V2)2-39
Request Interrupt (RI), (V2)2-40, (V2)2-44
Reserve Channel Service, (V1)6-153, (V1)7-156
Reserve Dual-Ported Disk/Set Single-Channel ACM Mode Service, (V1)6-145, (V1)7-148
Reserve FHD Port Service, (V1)6-220, (V1)7-212
Reset a Flag, (V2)1-88
Reset a Memory Word, (V2)2-32
Reset Option Lower Service, (V1)6-148, (V1)7-152
Resident Executive Services (H.REXS), (V1)5-49
Resident Shared Image
load, (V2)2-24
remove, (V2)2-22
Resource Control, (V1)4-4
Resource Create Block, see RCB
Resource Descriptor (RD)
allocation map, (V1)6-208, (V1)6-212
description, (V1)4-1, (V1)4-22, (V1)5-20, (V1)5-51
modify, (V1)6-112, (V1)7-113
modify user area, (V1)6-114, (V1)7-115
read, (V1)6-99, (V1)7-144
rewrite, (V1)6-146, (V1)7-150
rewrite user area, (V1)6-147, (V1)7-151
Resource Descriptor Tape Record, see RDTR
Resource Identifier (RID), (V1)4-31, (V1)5-10, (V1)5-55, L-66
Resource Inquiry Service, (V1)6-93, (V1)7-96
Resource Logging Block (RLB), (V1)6-105, (V1)7-108, L-67
Resource Management, (V1)3-1, (V1)4-1
dynamic, (V1)4-1
static, (V1)4-1
Resource Management Module, see H.REMM
Resource Requirement Summary (RRS), (V1)5-4, L-68
Resourcemark Lock Service, (V1)6-150, (V1)7-154
Resourcemark Table, increase size, (V3)7-49
Resourcemark Unlock Service, (V1)7-155
Resources
access, (V1)3-3
blocked I/O, (V1)3-4
execute channel program, (V1)3-3
logical device, (V1)3-4
logical file, (V1)3-4
access attributes
directories, (V1)3-9
files, (V1)3-10
memory partitions, (V1)3-10
volumes, (V1)3-8
allocate, (V1)5-1, (V1)5-2, (V1)5-3, (V1)6-8, (V1)7-12
dynamic, (V1)3-3
static, (V1)3-3, (V1)3-18
assign, (V1)5-1, (V1)5-2, (V1)5-3, (V1)5-4, (V1)6-8, (V1)7-12
attach, (V1)3-2, (V1)5-2
attributes, (V1)3-6
modify, (V1)3-5
protection, (V1)3-6
classes, (V1)4-2
conflicts, (V1)5-15

create, (V1)3-2
deadlock, (V1)4-50
deallocate, (V1)6-42, (V1)7-53
deassign, (V1)6-42, (V1)7-53
define, (V1)3-2
delete, (V1)3-2, (V1)6-48, (V1)7-57
dequeue, (V1)4-4
detach, (V1)3-4
directory structure, (V1)4-6
disk structure, (V1)4-6
display listing, (V2)3-44
enqueue, (V1)4-4
error handling, (V1)5-15
exclusive allocation, (V1)6-101, (V1)6-188, (V1)7-104, (V1)7-190
extension, (V1)3-11
functions, (V1)3-2
I/O, (V1)5-1
inquiry, (V1)3-5, (V1)6-93, (V1)7-96
log information, (V1)6-103, (V1)7-106, (V2)3-42, (V2)3-43, (V2)3-44
logical, (V1)3-1
modify attributes, (V1)3-5
multiprocessor, (V1)4-47
nonshareable, (V1)4-2
open, (V1)5-3, (V1)5-14, (V1)6-122, (V1)7-121
other, (V1)3-6, (V1)4-2, (V3)7-42
owner, (V1)3-6, (V1)4-2, (V3)7-43
pathnames, (V1)4-7
physical, (V1)3-1
print, (V1)5-38
project group, (V1)3-6, (V1)4-2, (V3)7-48
protection, (V1)4-13
punch, (V1)5-38
shareable, (V1)3-6, (V1)4-2
access control, (V1)4-5
exclusive, (V1)3-7, (V1)4-3
explicit, (V1)3-7, (V1)4-3
implicit, (V1)3-8, (V1)4-3
terminate operations, (V1)6-21, (V1)7-27
types, (V1)3-1, (V1)4-1
unformatted media, (V1)5-25
user classes, (V1)3-6, (V1)4-2
Restart, (V3)5-1
Restore files from tape, (V2)3-48, (V2)3-53
Restrictions, user, (V3)10-4
Resume Task Execution, (V1)6-167, (V1)7-171, (V2)2-40
Resume Task Execution Service, (V1)6-167, (V1)7-171
Return Pathname String, (V1)6-128, (V1)7-42
Rewind File Service, (V1)6-154, (V1)7-149
Rewind Magnetic Tape, (V2)3-53
Rewrite Descriptor Service, (V1)6-146, (V1)7-150
Rewrite Descriptor User Area Service, (V1)6-147, (V1)7-151
RID, (V1)4-31, (V1)5-10, (V1)5-55, L-66
RLB, (V1)6-105, (V1)7-108, L-67
RMSS, (V1)4-15, (V3)7-41
Root Directory, (V1)4-6
RRS, (V1)5-4, L-68
RTOM Interval Timer, (V3)7-32
Run Receiver, (V1)2-22, (V1)2-23
establish, (V1)2-23
exit, (V1)2-24, (V1)6-203, (V1)7-205
Run Request, (V1)1-11
end-action processing, (V1)2-26
exit, (V1)6-202, (V1)7-204
parameters, (V1)2-24, (V1)6-84, (V1)7-87
send to task, (V1)2-26, (V1)6-164, (V1)7-168
RXB, (V1)2-33, L-74

- S -

Save Files, (V2)3-54, (V2)3-57
display, (V2)3-43
Save Image, directory, (V2)3-4
Save Tape, (V2)3-1, (V2)3-2
SBO
change default device, (V2)2-56
delete file, (V2)2-14
logical file code assignment, (V2)1-22, (V2)1-36
output, redirect, (V2)2-37
repunch files, (V2)2-39
specify default device, (V3)7-45
Scan Terminal Input Buffer Service, (V1)6-178, (V1)7-180
Scanner Demo, (V4)2-59
Scheduler, select IPU/CPU, (V3)7-17
Scheduling
CPU, see CPU scheduling
I/O, (V1)2-42
IPU, ((V1)2-18
swap, see Swap Scheduling
task interrupt, (V1)2-20
Scratchpad, (V3)7-36
Screen Logic, TSM, (V2)1-21
Screen Width, (V2)1-75
SCSI Disk
device definition, (V3)7-13
media management, (V3)13-23
utility, (V4)2-50
SDT
master
boot from, (V3)2-12
contents, (V3)2-3
create, (V2)3-60

install, (V3)2-1
magnetic tape, (V3)2-10
utility tape, (V3)2-10
user
boot from, (V3)4-4
create, (V2)3-58, (V3)4-1
install, (V3)4-3
Search Memory for a Value, (V2)2-41
Search
within a file or files, (V4)2-38
within a source file, (V4)2-30, (V4)2-52
Security, (V3)10-22
limit logon time, (V3)10-23
limit terminal I/O inactivity, (V3)10-23
LOGONFLE examples, (V3)10-24
restrict logon attempts, (V3)10-22
Select Initial Input Source, (V2)1-93
Select Records
from a device, (V2)1-90
from a file, (V2)1-91
from initial input source, (V2)1-93
library format
from a device, (V2)1-92
from a file, (V2)1-93
Send a Message
to a task, (V1)6-162, (V1)7-166, (V2)2-42
to terminal users, (V2)1-101
to the console, (V1)6-187, (V1)7-189, (V3)7-11
to user's terminal, (V2)1-80, (V3)7-11
Send Message to Specified Task Service, (V1)6-162, (V1)7-166
Send Run Request to Specified Task Service, (V1)6-164, (V1)7-168
Sequential Access, (V1)4-35, (V1)5-38
Serial Printer Formatter/Spooler, (V4)2-51
Set Asynchronous Task Interrupt Service, (V1)6-10, (V1)7-14
Set Exception Handler Service, (V1)7-158
Set Exception Return Address Service, (V1)7-157
Set Exclusive File Lock Service, (V1)6-241
Set Exclusive Resource Lock Service, (V1)6-101, (V1)7-104
Set Flag
false, (V2)1-88
true, (V2)1-95
Set IPU Bias Service, (V1)6-98, (V1)7-102
Set Option Lower Service, (V1)6-163, (V1)7-167
Set Synchronization File Lock Service, (V1)6-238
Set Synchronous Resource Lock Service, (V1)6-157, (V1)7-161
Set Synchronous Task Interrupt Service, (V1)6-170, (V1)7-174
Set Timer, (V2)2-44
Set User Abort Receiver Address Service, (V1)6-166, (V1)7-170
Set User Status Word Service, (V1)6-155, (V1)7-159
SGO, logical file code assignment, (V2)1-22, (V2)1-36
Shadow Memory
access, (V3)10-57
allocate task space, (V2)1-98
assign by RRS, (V1)5-13
configurations, (V3)10-54, (V3)10-56
error messages, (V3)10-57
memory classes, (V3)10-56
overview, (V3)10-54
SYSGEN error messages, (V3)7-56
Shadow Utility, see J.SHAD
Share Memory with Another Task Service, (V1)6-250
Shared Image
access, (V1)4-46
attach, (V1)4-46
create, (V1)4-45
description, (V1)1-11, (V1)4-45
detach, (V1)4-46
exclude, (V1)7-73
include, (V1)7-93
load into memory, (V2)2-24
protection, (V1)4-45
remove from memory, (V2)2-22
Shared Memory, (V1)3-21, (V3)7-51
deallocate, (V1)6-233
dynamic partitions, (V1)3-18
multiprocessor, (V1)3-22
static partitions, (V1)3-18
SYSGEN error messages, (V3)7-51
Shared Memory Table (SMT), (V3)7-51
Shared Tasks, (V1)2-3
SHUTDOWN Macro
error messages, (V3)10-49
introduction, (V3)10-45, (V3)10-46
modify, (V3)10-47
usage, (V3)10-46
SID
change the default, (V2)2-56
specify default device, (V3)7-51
SJ.SWAPR2, (V3)10-49
SJ.XX.ER, (V3)10-18
SLO
change default device, (V2)2-56
delete file, (V2)2-13

logical file code assignment, (V2)1-22, (V2)1-36
output, (V2)1-103
redirect, (V2)2-37
reprint files, (V2)2-38
specify default device, (V3)7-33
SYSGEN, title, (V3)7-63
Small Computer System Interface, see SCSI Disk
SMAP, (V1)5-51, (V1)6-206, (V1)6-211
Software, unsupported, (V4)2-1
Software Interrupt System, (V1)1-7
Software Priorities, (V1)1-5
Sort/Merge, see FSORT2
Source Compare Program, (V4)2-25
Source Search Tool, (V4)2-30
Space Allocation, (V1)6-206
Special Keys, TSM, (V2)1-15
Split Image, see Extended MPX-32
Spool Batch Records
from a device, (V2)1-90
from a file, (V2)1-91
library format
from a device, (V2)1-92
from a file, (V2)1-93
Spooled Input
control, (V2)1-30
terminating conditions, (V2)1-32
Spooled Output, see SLO
Starter System, (V3)2-1
State Chain
head cell, (V1)2-12
queue, (V1)2-12
State Chain History, (V4)2-27
State Queues, (V1)2-13
Static Memory Allocation, (V1)3-18
Status Codes
H.REMM, (V1)5-17
H.VOMM, (V1)5-57
String Search, (V4)2-38
Submit Batch Job, (V2)1-45, (V2)1-102
Submit Batch Job on Boot-up, (V4)2-15
Submit Job from Disk File Service, (V1)6-226
Subroutine Libraries, (V1)1-21
Subroutine Library Editor, (V1)1-17
Suspend Task Execution Service, (V1)6-169, (V1)7-173
Suspend/Resume Service, (V1)6-168, (V1)7-172
SVC Type 1 Table, (V3)7-58
Swap Device, (V3)7-58
Swap File Size, (V3)7-46
Swap Monitor Program, (V4)2-54, (V4)2-55
Swap Parameter Modifier Program, (V4)2-53
Swap Quantum, (V3)7-59
Swap Scheduler
algorithms, (V3)10-50
call back swap-on priority only (CB.SOPO), (V3)10-51
description, (V3)10-49
errors, (V3)10-53
swap thrash control, (V3)10-51
task group outswap limits, (V3)10-52
user set inhibit flag (US.SWIF), (V3)10-51
user set swap-on priority only (US.SOPO), (V3)10-51
wait state ordering, (V3)10-50
wait state swap-on priority only (SOPO), (V3)10-51
Swap Scheduling, (V1)2-43
entry conditions, (V1)2-43
exit conditions, (V1)2-44
inswap process, (V1)2-45
outswap process, (V1)2-45
selection of inswap and outswap candidates, (V1)2-44
structure, (V1)2-43
Swapper Percentage Active Monitor, (V4)2-53
SYC
description, (V2)1-16
I/O input limitations, (V2)1-23
logical file code assignment, (V2)1-21, (V2)1-36
parameter replacement, (V2)1-30
terminal I/O, (V2)1-23
Symbol Table File Name, (V3)7-60
Symbolic Debugger/X32, (V1)1-18
Synchronization File Lock
release, (V1)6-237
set, (V1)6-238
Synchronized Access, (V1)6-157, (V1)6-190, (V1)7-161, (V1)7-193
SYSGEN, (V1)1-20, (V3)3-1, (V3)7-1
access, (V3)7-4
description, (V3)7-1
directive input file, (V3)3-1
directive summary, (V3)7-4
directives
ACTIVATE, (V3)7-10
AGE, (V3)7-10
ARTSIZE, (V3)7-11
BATCHMSG, (V3)7-11
BATCHPRI, (V3)7-11
BEGPGOUT, (V3)7-12
CDOTS, (V3)7-12
/CHANNELS, (V3)7-13
CMIMM, (V3)7-13
CMPMM, (V3)7-13
CONTROLLER, (V3)7-13
DBGFILE, (V3)7-16

DEBUGTLC, (V3)7-16
DELTA, (V3)7-17
DEMAND, (V3)7-17
DEVICE, (V3)7-18
DISP, (V3)7-26
DPTIMO, (V3)7-27
DPTRY, (V3)7-27
DTSAVE, (V3)7-27
//END, (V3)7-28
ENDPGOUT, (V3)7-28
EXTDMPX, (V1)3-28, (V3)7-28
/FILES, (V3)7-30
FLTSIZE, (V3)7-30
//HARDWARE, (V3)7-30
HELP, (V3)7-30
/INTERRUPTS, (V3)7-31
IOQPOOL, (V3)7-31
IPU, (V3)7-32
ITIM, (V3)7-32
ITLB, (V3)7-32
JOBS, (V3)7-32
KTIMO, (V3)7-33
LOD, (V3)7-33
LOGON, (V3)7-33
MACHINE, (V3)7-34
MAPOUT, (V3)7-35
MDT, (V3)7-35
/MEMORY, (V3)7-35
MMSG, (V3)7-36
MNWI, (V3)7-36
MODE, (V3)7-36
MODULE, (V3)7-37
/MODULES, (V3)7-38
MRUN, (V3)7-38
MSGPOOL, (V3)7-38
MTIM, (V3)7-39
NAME, (V3)7-39
NOANSI, (V3)7-40
NOBASE, (V3)7-40
NOCMS, (V3)7-40
NODEMAND, (V3)7-40
NOLACC, (V3)7-41
NOMAPOUT, (V3)7-41
NOSYSVOL, (V3)7-41
NOTDEF, (V3)7-41
NOTSMEXIT, (V3)7-42
NTIM, (V3)7-42
OTHERS, (V3)7-42
/OVERRIDE, (V3)7-43
OWNER, (V3)7-43
OWNERNAME, (V3)7-44
/PARAMETERS, (V3)7-44
/PARTITION, (V3)7-44
PASSWORD, (V3)7-44
PATCH, (V3)7-45
PCHFILE, (V3)7-45
POD, (V3)7-45
POOL, (V3)7-46
PRIORITY, (V3)7-46
PROGRAM, (V3)7-47
PROJECT, (V3)7-48
RLWU, (V3)7-48
/RMSTABLS, (V3)7-49
RMTSIZE, (V3)7-49
SAPASSWD, (V3)7-49
/SECURITY, (V3)7-49
SEQUENCE, (V3)7-50
SGOSIZE, (V3)7-50
SHARE, (V3)7-51
SID, (V3)7-51
SIZE, (V3)7-51
SMD, (V3)7-57
//SOFTWARE, (V3)7-57
SVC, (V3)7-58
SWAPDEV, (V3)7-58
SWAPLIM, (V3)7-59
SWAPSIZE, (V3)7-59
SWP, (V3)7-60
SYCSIZE, (V3)7-60
SYMTAB, (V3)7-60
/SYSDEVS, (V3)7-60
SYSMOD, (V3)7-61
SYSONLY, (V3)7-61
SYSTEM, (V3)7-61
SYSTRAP, (V3)7-62
/TABLES, (V3)7-62
TERMPRI, (V3)7-62
TIMER, (V3)7-62
TITLE, (V3)7-63
TQFULL, (V3)7-63
TQMIN, (V3)7-63
TRACE, (V3)7-64
/TRAPS, (V3)7-64
TSMEXIT, (V3)7-64
USERPROG, (V3)7-65
/VP, (V3)7-65
VP, (V3)7-65
VPID, (V3)7-65
logical file code summary, (V3)7-3
logical file codes, (V3)7-2
object input file, (V3)3-1
options, (V3)7-3
running, (V3)3-3
SYSINIT, (V3)2-23
System
build, (V3)1-1, (V3)3-1
maintenance, (V3)1-1
new default image, (V3)5-3
recovery, (V3)6-1
restart, (V3)5-1
shutdown, (V3)10-45
test, (V3)3-1, (V3)3-4

System Administrator, (V1)4-13
System Administrator Services
abort codes and messages, (V3)10-18
ACM/MFP initialization, (V3)10-35
ALIM initialization, (V3)10-26
ANSI tape label utility (J.LABEL), (V3)10-62
device initialization, (V3)10-19
floppy disk initialization (J.FORMF), (V3)10-42
INIT, (V3)10-20
introduction, (V3)10-1
job accounting (M.ACCNT), (V3)10-15
KEY program, see KEY
LOGONFLE, (V3)10-20
M.CNTRL, (V3)10-17
M.ERR and *xx*.ERR files, (V3)10-18
M.KEY, (V3)10-3
M.PRJCT, (V3)10-11
memory disk partition, (V3)10-58
multiprocessor recovery (J.UNLOCK), (V3)10-53
operator intervention inhibit, (V3)10-44
password control, (V3)7-49
PROJECT program, see PROJECT
security, (V3)10-22
shadow memory, (V3)10-54
swap scheduler control, (V3)10-49
system console messages, (V3)10-41
system date/time backup (J.DTSAVE), (V3)10-44
system date/time update, (V2)2-20
system shutdown, (V3)10-45
terminal initialization, (V3)10-19
volume mount (M.MOUNT), (V3)10-43
SYSTEM as Ownername, (V3)7-33, (V3)7-61
System Binary Output, see SBO
System Builder, (V3)2-23
System Console, (V2)2-4
configuration, (V3)2-1
device definition, (V3)7-21
device specification, (V1)5-28
messages, (V3)10-41
System Console Type Service, (V1)6-187, (V1)7-189
System Console Wait Service, (V1)6-41, (V1)7-51
System Control File, see SYC
System Date and Time, update, (V2)2-20
System Date/Time Backup Program (J.DTSAVE), (V3)10-44
System Debugger (H.DBUG1, H.DBUG2)
accessing, (V3)8-6
arithmetic operators, (V3)8-2
attach, (V2)2-12
base characters, (V3)8-4
bases, (V3)8-4
breakpoints, (V3)8-3
console address for stand-alone I/O, (V3)7-16
directive list example, (V3)8-32
directive summary, (V3)8-7
directives
AB (Absolute), (V3)8-9
AD (Address), (V3)8-9
AR (Arithmetic), (V3)8-10
AS (Assemble Instruction), (V3)8-10
BA (Base), (V3)8-10
BR (Breakpoint), (V3)8-11
BY (Bye), (V3)8-11
CB (Change Base Register), (V3)8-12
CD (Display Command List), (V3)8-12
CE (Zero Command List), (V3)8-12
CH (Display Controller Definition Table Entry), (V3)8-12
CL (Terminate Build Directive List Mode), (V3)8-13
CM (Change Memory), (V3)8-13
CO (Continue), (V3)8-13
CR (Change Register), (V3)8-14
CS (Build Directive List), (V3)8-14
CT (Continue then Terminate), (V3)8-14
CX (Execute Directive List), (V3)8-14
DB (Display Base Register), (V3)8-15
DE (Delete), (V3)8-15
DI (Display Instruction), (V3)8-15
DM (Display Memory), (V3)8-15
DQ (Display Dispatch Queue Entry), (V3)8-16
DR (Display Register), (V3)8-16
DS (Display Symbolic), (V3)8-16
DT (Display Event Trace), (V3)8-17
DU (Dump), (V3)8-17
EC (Echo), (V3)8-17
ET (Enter Event Trace Point), (V3)8-17
GO (Go), (V3)8-18
HC (Display Dispatch Queue Head Cell), (V3)8-18
LB (List Breakpoint), (V3)8-19
LP (Line Printer), (V3)8-19
LT (List Mobile Event Trace Point), (V3)8-19
MR (Map Register), (V3)8-19
MS (Modify CPU Scratchpad Location), (V3)8-20
PD (Display Patch List), (V3)8-20
PE (Zero Patch List), (V3)8-20
PR (Terminate Build Patch List Mode), (V3)8-21
PS (Program Status), (V3)8-21

PT (Build Patch List), (V3)8-21
PV (Convert Physical Address to Virtual Address), (V3)8-21
PX (Execute Patch List), (V3)8-21
RB (Reset Bases), (V3)8-22
RE (Remap), (V3)8-22
RT (Remove Event Trace Point), (V3)8-22
SE (Search Equivalent), (V3)8-23
SM (Set Mask), (V3)8-23
SP (Scratchpad Dump), (V3)8-23
SY (Symbolic), (V3)8-23
TB (Trace Back), (V3)8-24
TE (Terminate), (V3)8-24
TR (Trace), (V3)8-24
TS (Trace Stop), (V3)8-25
TY (Terminal), (V3)8-25
UD (Display Unit Definition Table Entry), (V3)8-25
VP (Convert Virtual Address to Physical Address), (V3)8-26
display a program, (V3)8-29
display memory, (V3)8-27
execution breakpoints, (V3)8-3
expressions, (V3)8-5
indirection, (V3)8-5
introduction, (V3)8-1
operator restrictions, (V3)8-5
parts of, (V3)8-1
patch list example, (V3)8-33
practice session, (V3)8-26
registers, (V3)8-5
special functions, (V3)8-3
special operators, (V3)8-2
System Dispatch Queue, display, (V2)2-26
System Distribution Tape, see SDT
System General Output, see SGO
System Generation, see SYSGEN
System Halt Analysis, (V3)6-2
System Image
build a new default, (V3)5-3
filename, (V3)7-61
System Input Device, see SID
System Listed Output, see SLO
System Modules
replace, (V3)7-61
specify, (V3)7-65
System Nonresident Media Mounting Task, (V1)1-13
System Output Queues, display, (V2)2-26
System Patch File, display, (V2)2-26
System Protection, (V3)3-5
System Recovery, (V3)6-1
System Restart, (V3)5-1
System Services
base mode, (V1)7-1
IPU executable, (V1)7-5
syntax rules, (V1)7-2
cross reference, B-1
H.VOMM macros, (V1)5-65
nonbase mode, (V1)6-1
IPU executable, (V1)6-3
overview, (V1)1-10
return conventions, (V1)5-16
status posting, (V1)5-16
System Shutdown
error messages, (V3)10-48
J.SHUTD, (V3)10-45
overview, (V3)10-45
SHUTDOWN, (V3)10-45, (V3)10-46
volume clean-up, (V3)10-46
System Start-up, Generation, and Installation, see SYSGEN
System Task Restrictions, (V2)2-4
System Volume, (V1)4-15, (V3)5-5

- T -

Tab Settings, TSM, (V2)1-20
Tabs, (V4)2-56
Tape Drive, device definition, (V3)7-23
Task
abort dump, (V3)7-36
abort, see Task, execution, abort
base nonshared, (V1)2-3
base shared, (V1)2-6
change priority, (V1)6-131, (V1)7-131
connect to interrupt level, (V1)6-35, (V1)7-41, (V2)2-10
create timer, (V1)6-159, (V1)7-163
delete timer, (V1)6-56, (V1)7-66, (V2)2-12
demand page, (V1)3-17
disconnect from interrupt level, (V2)2-15
display status, (V2)2-26, (V2)2-46
environment, (V1)6-66, (V1)7-72
execution, (V1)2-7
abort, (V1)2-46, (V1)6-16, (V1)6-50, (V1)7-21, (V1)7-59, (V2)1-19, (V2)2-6
return control, (V1)6-166, (V1)7-170
attach debugger, (V2)1-19, (V2)2-12
continue, (V2)1-19, (V2)1-20, (V2)1-49, (V2)2-11
delete task, (V2)1-20
from directive files, (V2)1-27, (V2)1-89
from system service, (V1)6-5, (V1)7-7

hold, (V1)6-87, (V1)7-90, (V2)1-19, (V2)2-23
kill, (V2)2-25
phase 1 of activation, (V1)2-7
phase 2 of activation, (V1)2-8
priorities, (V1)2-10
resume, (V1)6-167, (V1)6-185, (V1)7-171, (V1)7-187, (V2)2-10, (V2)2-40, (V2)2-44, (V2)2-58
suspend, (V1)6-41, (V1)6-169, (V1)7-51, (V1)7-173
suspend/resume, (V1)6-168, (V1)7-172
termination, (V1)2-46, (V1)2-47, (V1)6-69, (V1)7-75
TSA, (V1)2-8
under OPCOM, (V2)2-7, (V2)2-21, (V2)2-44
under TSM, (V2)1-16, (V2)1-34, (V2)1-45, (V2)1-59, (V2)1-88, (V2)1-102
exit status, (V1)7-78
hold, (V2)2-23
identification, (V1)2-1, (V1)6-88, (V1)7-91
interactive, characteristics, (V2)1-25
interrupt, (V2)1-19
context storage, (V1)2-21
level gating, (V1)2-21
levels, (V1)2-20
OPCOM, (V2)2-6, (V2)2-9, (V2)2-25
receivers, (V1)2-20
scheduling, (V1)2-20
system services (V1)2-20, (V1)6-19, (V1)6-55, (V1)6-59, (V1)6-64, (V1)6-97, (V1)6-170, (V1)6-171, (V1)6-198, (V1)7-24, (V1)7-63, (V1)7-67, (V1)7-70, (V1)7-101, (V1)7-174, (V1)7-175, (V1)7-200
system services summary, (V1)2-35
user break receivers, (V1)2-21
limits of base mode, (V1)7-103
multicopied, (V1)2-3
name, (V1)2-1
nonbase mode vs. base mode, (V1)2-2
nonbase nonshared, (V1)2-3, (V1)2-5
nonbase shared, (V1)2-6
number, (V1)2-1
obtain status, (V1)6-118, (V1)7-120
option word inquiry, (V1)6-126, (V1)6-127, (V1)7-124, (V1)7-125
override parameters, (V1)6-133, (V1)7-133
priorities, (V1)2-10
priority levels, (V1)1-7
privileged, (V1)6-132, (V1)7-132
shared, (V1)2-3
specify user name, (V1)6-253
state, (V1)2-12, (V1)2-13, (V2)1-20, (V2)2-28, (V2)2-52
structure, (V1)2-2
swap scheduling, (V3)10-49
termination, see Task, execution, termination
test timer, (V1)6-184, (V1)7-186
unique, (V1)2-3
unprivileged, (V1)6-192, (V1)7-192
Task Cataloger, see CATALOG
Task CPU Execution Time Service, (V1)6-204, (V1)7-206
Task Debugger, (V1)1-16
Task Identification, (V1)2-1
Task Interrupt, see Task, interrupt
Task Name, (V1)2-1, (V2)2-4
Task Number, (V1)2-1, (V2)2-4
Task Option Doubleword Inquiry Service, (V1)6-126, (V1)7-124
Task Option Word Inquiry Service, (V1)6-127, (V1)7-125
Task Service Area, see TSA
Task Structure, (V1)2-2
Task-Synchronized Access to Common Resources, (V1)2-49
TCPB, (V1)5-41, L-75
TCW, (V1)5-37
TDEFLIST File, (V2)11-5
Temporary Files, see File, temporary
TERMDEF
access with M.GETDEF, (V1)6-81, (V2)11-18
access with M_GETDEF, (V1)7-81
components
J.TDEFI, (V2)11-4
J.TSET, (V2)11-21
LOGONFLE, (V2)11-17
M.GETDEF, (V2)11-18
TDEFLIST, (V2)11-5
TERMDEF file, (V2)11-8
TERMPART, (V2)11-3
demo, (V4)2-28, (V4)2-31
exclude support, (V3)7-41
file
booleans, (V2)11-9
control strings, (V2)11-9
cursor addressing, (V2)11-10
sample, (V2)11-12
illustration, (V2)11-2
information block, (V2)11-18
utility, (V4)2-19
Terminal Definition Facility, see TERMDEF
Terminal I/O
close, (V2)1-24

example session, (V2)1-108
exit, (V2)1-14, (V2)1-59
input limitations, (V2)1-23
open, (V2)1-24
reads, (V2)1-23
rewind scanner, (V2)1-24
scan input buffer, (V1)6-178, (V1)7-180
SYC, (V2)1-23
type identification, (V2)11-8, (V2)11-17, (V2)11-21
wait state, (V1)6-195, (V1)7-197, (V2)1-105, (V2)2-60
wakeup, (V2)1-8, (V2)1-20
writes, (V2)1-24
Terminal Initialization, (V3)3-5, (V3)10-19
Terminal Initializer/Loader, (V4)2-56
Terminal Services Manager, see TSM
Terminal Timeout Counter, (V3)10-23
Terminal, user, see UT
Terminate a Directive File, (V2)1-58
Terminate Processing, (V2)1-49
Terminate Task Execution Service, (V1)6-69, (V1)7-75
TERMPART, (V2)11-3
Test Timer Entry Service, (V1)6-184, (V1)7-186
Test User Status Word Service, (V1)6-183, (V1)7-185
Testing a SYSGENed System, (V3)3-4
Text Editor, (V1)1-17
Time and Date Formats, H-1
Time Instruction Sequences, (V4)2-38
Time Management, (V1)1-12
Time-of-Day Inquiry Service, (V1)6-172, (V1)7-176
Time Quantum
controls, (V1)2-11
maximum, (V3)7-63
minimum, (V3)7-63
Time Units, (V3)7-39, (V3)7-42
Timer Entry
create, (V1)6-159, (V1)7-163
delete, (V1)6-56, (V1)7-66
test, (V1)6-184, (V1)7-186
Timer Scheduler, (V1)1-12
Timer Table Entries, (V3)7-62
Trap Handlers, (V1)2-51, (V3)7-47
override defaults, (V3)7-62
Trap Online User's Task Service, (V1)6-171, (V1)7-175
Trap Processors, (V1)1-12, (V1)2-51, (V3)7-47
Truncate File Service, (V1)6-177, (V1)7-179
TSA
description, (V1)2-8
extended (V1)3-34
move, (V1)3-28, (V1)3-32, (V2)1-60
pushdown stack area, (V1)2-10
starting address, (V1)6-86, (V1)7-89
structure, (V1)2-10
TSM, (V1)1-13, (V2)1-1
break key, (V2)1-19
break processor, (V2)1-27
case sensitivity, (V2)1-20
conditional processing, (V2)1-29
device initialization, (V3)10-19
directive syntax, (V2)1-33
directives
$$, (V2)1-106
$$$, (V2)1-106
$ACCOUNT, (V2)1-33
$ACTIVATE, (V2)1-34
$ALLOCATE, (V2)1-35
$ASSIGN, (V2)1-36
$ASSIGN1, (V2)1-41
$ASSIGN2, (V2)1-42
$ASSIGN3, (V2)1-43
$ASSIGN4, (V2)1-44
$BATCH, (V2)1-45
$CALL, (V2)1-46
$CHANGE, (V2)1-48
$CLEAR, (V2)1-49
$CONTINUE, (V2)1-49
$CREATE, (V2)1-50
$DEBUG, (V2)1-50
$DEFM, (V2)1-51
$DEFNAME, (V2)1-52
$DELETE, (V2)1-53
$DIRECTORY, (V1)6-179, (V1)7-181
$DISABLE (V2)1-54
$DISMOUNT, (V2)1-55
$ENABLE (V2)1-57
$END, (V2)1-58
$ENDM, (V2)1-58
$EOJ, (V2)1-58
$ERR, (V2)1-58
$EXECUTE, (V2)1-59
$EXIT, (V2)1-59
$EXTDMPX, (V1)3-33, (V2)1-60
$GETPARM, (V1)6-181
$GOBACK, (V2)1-62
$GOTO, (V2)1-63
$IFA, (V2)1-64
$IFF, (V2)1-66
$IFP, (V2)1-64
$IFT, (V2)1-69
$INIT PRO, (V2)10-7
$JOB, (V2)1-72
$LINESIZE, (V2)1-75
$LIST, (V2)1-76
$MAPOUT, (V2)1-76
$MOUNT, (V2)1-77

$NOMAPOUT, (V2)1-79
$NOTE, (V2)1-80
$OBJECT, (V2)1-80
$OPTION, (V2)1-81
$PAGESIZE, (V2)1-83
$PRINT, (V2)1-84
$PROJECT, (V1)6-179, (V1)7-181
$RECALL, (V2)1-85
$REMOVE, (V2)1-86
$RENAME, (V2)1-87
$RESETF, (V2)1-88
$RESTART, (V3)5-2
$RRS, (V1)6-181
$RUN, (V2)1-88
$SELECT, (V2)1-89
$SELECTD, (V2)1-90
$SELECTF, (V2)1-91
$SELECTLD, (V2)1-92
$SELECTLF, (V2)1-93
$SELECTS, (V2)1-93
$SET, (V2)1-94
$SETF, (V2)1-95
$SETI, (V2)1-96
$SHADOW, (V2)1-98
$SHOW, (V2)1-99
$SIGNAL, (V2)1-101
$SPACE, (V2)1-101
$SUBMIT, (V2)1-102
$SYSOUT, (V2)1-103
$TABS, (V1)6-179, (V1)7-181
$URGENT, (V2)1-104
$USERNAME, (V2)1-104
$VOLUME, (V1)6-179, (V1)7-181
$WAIT, (V2)1-105
$WHO, (V2)1-106
%label, (V2)1-52
exit, (V2)1-59
exit when inactive, (V3)7-42
interactive tasks, (V2)1-25
introduction, (V2)1-1
JCL directive summary, (V2)1-2
macro looping, (V2)1-29
options, (V2)1-17, (V2)1-24, (V2)1-81
abort, (V2)1-18, (V2)1-81
clear, (V2)1-17, (V2)1-81
command, (V2)1-17, (V2)1-81
cpuonly, (V2)1-18, (V2)1-81
dump, (V2)1-19, (V2)1-81
error, (V2)1-18, (V2)1-81
ipubias, (V2)1-19, (V2)1-81
l/c, (V2)1-19, (V2)1-81
lower, (V1)6-148, (V1)6-163, (V1)7-152, (V1)7-167, (V2)1-17, (V2)1-81
noabort, (V2)1-18, (V2)1-81
nocommand, (V2)1-17, (V2)1-81
noerror, (V2)1-18, (V2)1-81
nowrap, (V2)1-18, (V2)1-81
prompt, (V2)1-17, (V2)1-81
quiet, (V2)1-18, (V2)1-81
retain, (V2)1-17, (V2)1-81
text, (V2)1-17, (V2)1-81
u/c, (V2)1-19, (V2)1-81
unquiet, (V2)1-18, (V2)1-81
wrap, (V2)1-18, (V2)1-81
parameter passing, (V2)1-29
procedure call block, (V1)6-180, (V1)7-182, L-50
procedure call directive strings, (V1)6-179, (V1)7-181
scanner, (V2)1-25
screen logic, (V2)1-21
set options, (V2)1-81
special keys, (V2)1-15
tab settings, (V2)1-20
TSM Procedure Call Block (PCB), (V1)6-180, (V1)7-182, L-50
TSM Procedure Call Service, (V1)6-179, (V1)7-181
TSM Scanner Demo, (V4)2-59
Type Control Parameter Block (TCPB), (V1)5-41, L-75

- U -

UDT, (V2)11-4, (V2)11-17, (V2)11-21, L-77
UDT Display, (V4)2-60
Unformatted Media, (V1)5-25
Unique Tasks, (V1)2-3
Unit Definition Table (UDT), (V2)11-4, (V2)11-17, (V2)11-21, L-77
Unlock and Dequeue Shared Memory Service, (V1)6-252
Unsupported Software, (V4)2-1
UPDATE, (V1)1-19
Upspace Service, (V1)6-193, (V1)7-195
User Context
overwrite, (V1)7-138
store values, (V1)7-80
User Modules, (V3)7-37
User Name, (V1)6-253
User Name Specification Service, (V1)6-253
User Status Word, (V1)6-155, (V1)6-183, (V1)7-159, (V1)7-185
User Terminal, see UT
User Volume, (V1)4-15
UT
device definition, (V3)7-21
exit, (V2)1-59
I/O input limitations, (V2)1-23
logical file code assignment, (V2)1-21
wait state, (V2)1-105, (V2)2-60
wakeup, (V2)1-8, (V2)1-20, (V3)10-20

Utilities
AIDDB, (V1)1-16
ASMX32, (V1)1-17
CATALOG, see CATALOG
DEBUGX32, (V1)1-18
DPEDIT, (V1)1-17
EDIT, (V1)1-17
illustration, (V1)1-3
J.VFMT, see J.VFMT
KEY, see KEY
LIBED, (V1)1-17
LINKX32, (V1)1-18
MACLIBR, (V1)1-16
MACX32, (V1)1-17
MEDIA, (V1)1-19
OBJX32, (V1)1-18
SYSGEN, see SYSGEN
UPDATE, (V1)1-19
VOLMGR, see VOLMGR

- V -

Validate Address Range Service, (V1)6-194, (V1)7-196
Vector Processor Configuration, (V3)7-65, (V3)7-66
VOLMGR, (V1)1-17, (V2)3-1
access, (V2)3-14
BRIEF option, (V2)3-8
clear options, (V2)3-23
directive summary, (V2)3-7
directive syntax, (V2)3-18
directives
BACKSPACE FILE, (V2)3-22
BACKSPACE IMAGE, (V2)3-22
CLEAR, (V2)3-23
CONVERT, (V2)3-23
COPY, (V2)3-25
CREATE COMMON, (V2)3-29
CREATE DIRECTORY, (V2)3-32
CREATE FILE, (V2)3-34
DELETE COMMON, (V2)3-37
DELETE DIRECTORY, (V2)3-38
DELETE FILE, (V2)3-39
EXIT, (V2)3-40
EXTEND, (V2)3-41
HELP, (V2)3-41
LOG FILE, (V2)3-42
LOG IMAGE, (V2)3-43
LOG RESOURCE, (V2)3-44
LOG SAVEFILE, (V2)3-46
RENAME, (V2)3-47
RESTORE DIRECTORY, (V2)3-48
RESTORE POSITION, (V2)3-53
REWIND, (V2)3-53
SAVE, (V2)3-54
SAVE INCREMENTAL, (V2)3-57
SDT, (V2)3-58
SDT MASTER, (V2)3-60
SET, (V2)3-61
SKIP END, (V2)3-64
SKIP FILE, (V2)3-64
SKIP IMAGE, (V2)3-65
TRUNCATE, (V2)3-65
errors, (V2)3-66
global options, (V2)3-18, (V2)3-19
help, (V2)3-41
introduction, (V2)3-1
local options, (V2)3-18, (V2)3-19
logical file code assignments, (V2)3-16
options, (V2)3-17
BRIEF, (V2)3-8
clear, (V2)3-23
global, (V2)3-18, (V2)3-19
local, (V2)3-18, (V2)3-19
set, (V2)3-61
time, (V2)3-20
resource descriptor tape record, (V2)3-6
save image, (V2)3-3
save tape format, (V2)3-1
save tape structure, (V2)3-2
set options, (V2)3-61
temporary files, (V2)3-17
time options, (V2)3-20
wild card characters, (V2)3-15, (V2)3-17
Volume, (V1)4-14
access attributes, (V1)3-8, (V1)4-16
automatic public mount, (V3)9-11
boot block, (V1)4-22
descriptor, (V1)4-22
dismount, (V1)4-14, (V1)4-18, (V1)6-57, (V1)7-64, (V2)1-55, (V2)2-16
explicit, (V1)4-19
implicit, (V1)4-19
logical, (V1)4-18
physical, (V1)4-19
formatted, (V1)4-14
mount, (V1)4-14, (V1)4-17, (V1)6-115, (V1)7-116, (V2)1-77, (V2)2-33
explicit, (V1)4-18
implicit, (V1)4-18
logical, (V1)4-18
physical, (V1)4-17
multiprocessor, (V1)4-16, (V1)4-47, (V1)4-49
nonpublic, (V1)4-16
public, (V1)4-16, (V1)4-20
inhibit dismount, (V3)7-13
rename, (V4)2-44
resource access, (V1)5-20
space management, (V1)5-20
status, display, (V2)2-46
structure, (V1)4-20, (V1)4-21

system, (V1)4-15, (V1)4-20
types, (V1)4-15
user, (V1)4-15
user default, (V3)10-4
Volume Compress, see J.DSCMP
Volume Formatter, see J.VFMT
Volume Management Module, see H.VOMM
Volume Manager, see VOLMGR
Volume Resource Management, (V1)4-1

- **W** -

Wait for Any Break Interrupt Service, (V1)6-7, (V1)7-11
Wait for Any Message Interrupt Service, (V1)6-7, (V1)7-11
Wait for Any No-Wait Operation Complete Service, (V1)6-7, (V1)7-11
Wait I/O, (V1)5-32
Wait I/O Service, (V1)6-195, (V1)7-197
Wait States, (V1)2-25, (V1)2-26, (V1)6-63, (V1)6-195, (V1)7-11, (V1)7-15, (V1)7-197, (V2)1-105, (V2)2-60, (V3)10-50, (V3)10-51
Wakeup, (V2)1-8, (V2)1-20, (V3)10-20
Word Locations, dump, (V2)2-18, (V2)2-45
Write EOF Service, (V1)6-196, (V1)7-199
Write Record Service, (V1)6-197, (V1)7-198
xx.ERR Files, (V3)10-18

MPX-32™ Overview & System Services
Reference Manual
No. 323-001551-600

READER'S COMMENT FORM

Please use this form to communicate your views about this manual. The form is preaddressed and stamped for your convenience.

I rate this manual's:	Excellent	Good	Fair	Poor
Accuracy	____	____	____	____
Clarity	____	____	____	____
Completeness	____	____	____	____
Examples	____	____	____	____
Figures	____	____	____	____
Index	____	____	____	____
Organization	____	____	____	____
Retrievability of Information	____	____	____	____

Additional comments:

_____

_____

_____

_____

_____

_____

If you wish a reply, please print your name and mailing address:

_____

_____

_____

_____

What is your occupation/title? _____

_____

Thank you for your cooperation.

Note: Copies of Encore publications are available through your Encore representative or the customer service office serving your locality.

FOLD HERE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 2356 FORT LAUDERDALE, FL

POSTAGE WILL BE PAID BY ADDRESEE

ENCORE COMPUTER CORPORATION
ATTENTION: DOCUMENTATION COORDINATOR
6901 W. SUNRISE BLVD.
P.O. BOX 409148
FT. LAUDERDALE, FL 33340-9970

FOLD HERE

PLEASE TAPE DO NOT STAPLE

CUT ALONG LINE