

*System
Software
Operation
Guide*

Volume 1

B 1000 Systems

*(Relative to Mark 12.0 System Software Release)
Copyright © 1985, Burroughs Corporation, Detroit, Michigan 48232*

Burroughs cannot accept any financial or other responsibilities that may be the result of your use of this information or software material, including direct, indirect, special or consequential damages. There are no warranties extended or granted by this document or software material.

You should be very careful to ensure that the use of this software material and/or information complies with the laws, rules, and regulations of the jurisdictions with respect to which it is used.

The information contained herein is subject to change without notice. Revisions may be issued to advise of such changes and/or additions.

Comments or suggestions regarding this document should be submitted on a Field Communication Form (FCF) with the CLASS specified as 2 (S.SW: System Software), the Type specified as 3 (DOC), and the product specified as the 7-digit form number of the manual (for example, 1169000).

LIST OF EFFECTIVE PAGES

Page	Issue
Title	Original
ii	Original
iii	Original
iv	Blank
v thru xvii	Original
xviii	Blank
xix thru xx	Original
1-1 thru 1-9	Original
1-10	Blank
2-1 thru 2-11	Original
2-12	Blank
3-1 thru 3-17	Original
3-18	Blank
4-1 thru 4-176	Original
5-1 thru 5-169	Original
5-170	Blank
6-1 thru 6-21	Original
6-22	Blank
7-1 thru 7-3	Original
7-4	Blank
8-1 thru 8-17	Original
8-18	Blank
A-1 thru A-11	Original
A-12	Blank
B-1 thru B-9	Original
B-10	Blank
C-1 thru C-11	Original
C-12	Blank
D-1 thru D-11	Original
D-12	Blank
E-1 thru E-4	Original
1 thru 20	Original

TABLE OF CONTENTS

Section	Title	Page
	INTRODUCTION	xix
	Purpose of Manual	xix
	Organization of Manual	xix
	Related Documentation	xx
1	INTRODUCTION TO THE B 1000 SYSTEM	1-1
	The Master Control Program (MCP)	1-1
	System Description	1-1
	Device Requirements	1-1
	Central Service Module	1-2
	Interpreters	1-2
	Task Schedules and the System Mix	1-2
	Waiting Schedule	1-3
	FS System Command	1-3
	HW System Command	1-3
	RS System Command	1-3
	SP System Command	1-3
	WS System Command	1-3
	Active Schedule	1-3
	HS System Command	1-3
	JS System Command	1-3
	RS System Command	1-3
	SP System Command	1-3
	WS System Command	1-3
	System Mix	1-4
	DP System Command	1-4
	DS System Command	1-4
	ML System Command	1-4
	MX System Command	1-4
	Control Instructions	1-4
	Sources of Control Instructions	1-5
	Punched Cards	1-5
	Operator Display Terminal (ODT)	1-6
	ZIP Statement	1-6
	Generic Terms	1-6
	file-identifier	1-6
	file-title	1-7
	file-name	1-7
	program-name	1-8
	compiler-name	1-8
	interpreter-name	1-8
	unit-mnemonic	1-8
	SYSTEM disk	1-8
	user disk	1-8
	mix number	1-9

TABLE OF CONTENTS (Cont)

Section	Title	Page
2	OVERVIEW OF CONTROL INSTRUCTIONS	2-1
	Interrogating and Changing the Time and Date	2-1
	Interrogating the Status of Programs in the Mix	2-1
	Responding to Program Messages	2-2
	Status Checking	2-2
	Messages Concerning Program Status	2-4
	Disk Operation and Interrogate Messages	2-5
	Handling the System Options	2-6
	Handling Memory Dumps	2-6
	Handling Magnetic Tape Devices	2-6
	Handling Peripherals	2-7
	Changing the Status of Programs Running in Mix	2-8
	Performing Hexadecimal Conversion and Calculations	2-8
	Controlling Programs in the Schedule	2-8
	Controlling Backup (Spooling) Functions	2-9
	Control of Pseudo Card Readers	2-10
	Handling the Log Files	2-10
	Changing the System Software Environment	2-11
	Burroughs Network Architecture (BNA) Commands	2-11
	Work Flow Language (WFL) Command	2-11
3	INITIALIZING THE SYSTEM	3-1
	Hardware Initialization	3-1
	How to Create the System Cassettes	3-1
	Creating the clear/start Cassette	3-1
	Stand-alone Utility Cassettes (SDL Utility Programs)	3-2
	How to Initialize a Disk Pack	3-2
	How to Initialize the Operating System	3-3
	How to Coldstart the Operating System	3-3
	Using the COLDSTART/TAPE Cassette	3-3
	Using the COLDSTART/DISK Cassette	3-4
	How to clear/start the Operating System	3-6
	Temporary Operating Environment Changes	3-6
	General Specifications	3-7
	How to Set the Operating System Options	3-7
	Date and Time	3-7
	MCP Options	3-8
	AMCS	3-8
	BOJ	3-8
	BREL	3-8
	CHRG	3-9
	CLOS	3-9
	COPY	3-9
	DATE	3-9
	DEBUG	3-9
	DISP	3-9
	DUMP	3-9

TABLE OF CONTENTS (Cont)

Section	Title	Page
	EOJ	3-9
	FLMP	3-10
	LAB	3-10
	LIB	3-10
	LOG	3-10
	MEM	3-10
	MPRI	3-10
	ODTL	3-11
	OPEN	3-11
	PBD	3-11
	PBT	3-11
	RMOV	3-11
	RMSG	3-11
	SCHM	3-11
	SQRM	3-12
	TERM	3-12
	THR	3-12
	TIME	3-12
	TRMD	3-12
	VLCP	3-12
	VLIO	3-12
	WFL	3-13
	ZIP	3-13
	Installing New System Software Updates	3-13
	Determine Current Operating Environment	3-13
	Copy the Software to be Replaced	3-14
	Use Experimental Entries	3-14
	Loading a New SDL2 Interpreter	3-14
	Loading a New SDL2 Intrinsic File	3-15
	clear/start With OLD/Software	3-15
	Remove Old System Files	3-15
	Copy New System Files	3-15
	Perform the CM(s)	3-16
	SYSTEM/COPY and SDL2 Intrinsic File	3-16
	Loading SYSTEM/COPY	3-16
	Loading SDL2 Intrinsic File	3-16
	NDL Handler	3-17
	Clear/Start the New System Software	3-17
	Remove Old Software	3-17
	Experimental Name Table Entries	3-17
4	PROGRAM CONTROL INSTRUCTIONS	4-1
	AC	4-1
	AFTER	4-2
	AFTER.NUMBER	4-3
	AX	4-4
	CHARGE	4-5
	COMPILE	4-6

TABLE OF CONTENTS (Cont)

Section	Title	Page
	CONDITIONAL	4-8
	DATA	4-9
	DYNAMIC	4-10
	DYNAMIC.SPACES	4-12
	END	4-14
	ENDCTL	4-16
	EXECUTE	4-17
	FILE	4-18
	File Attributes	4-19
	ALLOCATE.AT.OPEN	4-20
	AREAS	4-21
	ASCII	4-22
	AUDITED	4-23
	AUTOPRINT	4-24
	BACKUP	4-25
	BACKUP.DISK	4-26
	BACKUP.TAPE	4-27
	BCL	4-28
	BINARY	4-29
	BLOCKS.PER.AREA	4-30
	BUFFERS	4-31
	CARD.PUNCH	4-32
	CARD.READER	4-33
	CASSETTE	4-34
	COPY	4-35
	DATA.RECORDER.80	4-36
	DEFAULT	4-37
	DELAYED.RANDOM	4-38
	DISK	4-39
	DISK.CARTRIDGE	4-40
	DISK.FILE	4-41
	DISK.PACK	4-42
	DRIVE	4-43
	DUMMY.FILE	4-44
	EBCDIC	4-45
	END.OF.PAGE	4-46
	EVEN.PARITY	4-47
	EXTEND	4-48
	FILE.TYPE	4-49
	FLEXIBLE	4-51
	FOOTING	4-52
	FORMS	4-53
	HARDWARE	4-54
	HEADER	4-55
	HOSTNAME	4-56
	IMPLIED.OPEN.DENIAL.FLAG	4-57
	INPUT	4-58

TABLE OF CONTENTS (Cont)

Section	Title	Page
	INTNAME	4-59
	INVALID.CHARACTERS	4-60
	LABEL.TYPE	4-61
	LINEFORMAT	4-62
	LOCK	4-63
	LOWER.MARGIN	4-64
	MAXIMUM.BLOCK.SIZE	4-65
	MAXRECSIZE	4-66
	MAXSUBFILES	4-67
	MINRECSIZE	4-68
	MULTI.PACK	4-69
	MY.NAME	4-70
	NAME	4-71
	NEW.FILE	4-72
	NOT	4-73
	NUMBER.OF.REMOTE.STATIONS	4-74
	ODD	4-75
	OPEN.LOCK	4-76
	OPEN.LOCKOUT	4-77
	OPEN.ON.BEHALF.OF	4-78
	OPTIONAL	4-79
	OUTPUT	4-80
	PACK.ID	4-81
	PAGE.SIZE	4-82
	PAPER.TAPE.PUNCH	4-83
	PAPER.TAPE.READER	4-84
	PORT.FILE	4-85
	PORT.KEY	4-86
	PRINTER	4-87
	PROTECTION	4-88
	PROTOCOL	4-89
	PSEUDO	4-90
	Q.FAMILY.SIZE	4-91
	Q.MAX.MESSAGES	4-92
	QUEUE	4-93
	RANDOM	4-94
	RECORDS.PER.BLOCK	4-95
	RECORD.SIZE.IN.BYTES	4-96
	READER.PUNCH.PRINTER	4-97
	READER.SORTER	4-98
	READER.SORTER.STATIONS	4-99
	READER.96	4-100
	REEL.NUMBER	4-101
	REMOTE	4-102
	REPETITIONS	4-103
	REVERSE	4-104
	REWIND	4-105

TABLE OF CONTENTS (Cont)

Section	Title	Page
	SAVE	4-106
	SECURITYTYPE	4-107
	SECURITYUSE	4-108
	SEND.ALL.ATTRIBUTES	4-109
	SEQUENTIAL	4-110
	SERIAL	4-111
	SIMPLE.HEADERS	4-112
	SPECIAL.FORMS	4-113
	STATIONS	4-114
	TAPE	4-115
	TAPE.NRZ	4-116
	TAPE.PE	4-117
	TAPE.7	4-118
	TAPE.9	4-119
	TITLE	4-120
	TRANSLATE	4-121
	TRANSLATE.FILE.NAME	4-122
	UNIT.NAME	4-123
	UNLABELED	4-124
	UPPER.MARGIN	4-125
	USER.BACKUP.NAME	4-126
	VARIABLE.BLOCK	4-127
	WITH.INTERPRET	4-128
	WITH.PRINT	4-129
	WITH.PUNCH	4-130
	WITH.STACKERS	4-131
	WORK.FILE	4-132
	FREEZE	4-133
	HOLD	4-134
	INTERPRETER	4-135
	INTRIN.DIRECTORY	4-136
	INTRINSIC.NAME	4-137
	INVISIBLE	4-138
	LEVEL	4-139
	MAXWAIT	4-140
	MEMORY	4-141
	MEMORY.PRIORITY	4-142
	MEMORY.STATIC	4-144
	MODIFY	4-145
	NO.DEATH.IN.FAMILY	4-146
	OBJ	4-147
	OVERRIDE	4-148
	PRIORITY	4-150
	PROCESSOR.PRIORITY	4-152
	PROTECTED	4-154
	RR	4-155

TABLE OF CONTENTS (Cont)

Section	Title	Page
	RUN	4-157
	START	4-158
	SCHEDULE.PRIORITY	4-159
	SECONDS.BEFORE.DECAY	4-161
	STREAM	4-163
	SWITCH	4-164
	SYMBOLIC.QUEUE.NAME	4-166
	TERMINATE	4-167
	THEN	4-168
	TIME	4-169
	UNCONDITIONAL.EXECUTION	4-171
	UNFREEZE	4-173
	UNOVERRIDE	4-174
	VIRTUAL.DISK	4-175
5	SYSTEM COMMANDS	5-1
	Command Entry	5-1
	Syntax Diagrams	5-1
	AB (AUTOBACKUP)	5-1
	AC (Response to Accept Message)	5-5
	ADD	5-6
	AL (Align Printer)	5-7
	AP (Auto Print)	5-8
	AT	5-9
	AX (Response to Accept Message)	5-10
	BB (Backup Blocks per Area)	5-11
	BD (Backup Designate)	5-12
	BF (Display Backup Files)	5-13
	CC (Control Card)	5-15
	CD (List Card Decks in Pseudo Readers)	5-16
	CHANGE	5-17
	CL (Clear Unit)	5-19
	CM (Change System Software)	5-20
	COPY	5-21
	CP (Compute)	5-22
	CQ (Clear Queue)	5-25
	CT (Chip Table)	5-26
	CU (Core Usage)	5-27
	DB (Data Base)	5-29
	DF (Date of File)	5-30
	DIR	5-31
	DL (Disk Location)	5-32
	DM (Dump Memory and Continue)	5-33
	DP (Dump Memory and Discontinue)	5-34
	DR (Change MCP Date)	5-35
	DS (Discontinue Program)	5-36
	DT (Change MCP Date)	5-37
	ED (Eliminate Pseudo Deck)	5-38

TABLE OF CONTENTS (Cont)

Section	Title	Page
	EM (ELOG Message)	5-39
	ER (Error Rate)	5-40
	ET (ELOG Transfer)	5-41
	FM (Response to Special Forms)	5-42
	FN (Display File Name)	5-43
	FR (Final Reel of Unlabeled Tape File)	5-44
	FS (Force from Waiting Schedule)	5-45
	GO (Resume Stopped Program)	5-46
	HALT	5-47
	HN (Hostname)	5-48
	HS (Hold in Waiting Schedule)	5-49
	HW (Hold in Waiting Schedule until Job EOJ)	5-50
	IB (Instruction Block)	5-51
	IC (Interpreter Count)	5-52
	IL (Ignore Label)	5-53
	IV (Invisible)	5-55
	JOBSTART	5-56
	JS (Jiggle Schedule)	5-57
	KA (Analyze Disk Directory)	5-58
	KB (Print ODTLOG)	5-60
	KC (Print Disk Segments in Character Format)	5-64
	KP (Print Disk Segments in Hexadecimal Format)	5-66
	LC (Log Comment)	5-68
	LD (Pseudo Load)	5-69
	LG (Transfer and Print Log)	5-71
	LN (Transfer and Print Log)	5-72
	LP (Lock Protection)	5-73
	LT (Load Translator)	5-74
	B 1247 Train Printer Control (Control Identification = @10@)	5-74
	B 1247-4 Train Printer Control (Control Identification = @3E@)	5-75
	MH (Modify Header)	5-77
	ML (Mix Limit)	5-79
	MM	5-80
	MP (Memory Priority)	5-82
	MR (Close Output File with Purge)	5-83
	MU (List Multipack File Tables)	5-84
	MX (Mix and Status Interrogation)	5-85
	NC (Network Controller)	5-86
	NET (Network Mode Change or Inquiry)	5-87
	NW (Network Message)	5-89
	OF (Optional File Response)	5-90
	OK (Continue Processing)	5-91
	OL (Display Peripheral Status)	5-92
	OU (Specify Output Device)	5-94
	PASSWORD	5-95
	PB (Print/Punch Backup)	5-96
	PD (Display Directory)	5-101

TABLE OF CONTENTS (Cont)

Section	Title	Page
	PF (Print Fetch)	5-103
	PG (Purge)	5-104
	PM (Print Memory Dump)	5-105
	PO (Power Off)	5-107
	PP (Processor Priority)	5-108
	PR (Change Priority)	5-109
	QF (Query File)	5-110
	QP (Query Program)	5-111
	RB (Remove Backup Files)	5-112
	RC (Recover Database)	5-114
	RD (Remove Pseudo Card Files)	5-115
	REMOVE	5-116
	RF (Remove Backup Files)	5-118
	RL (Relabel User Disk)	5-120
	RM (Remove Duplicate Disk File)	5-121
	RN (Assign Pseudo Readers)	5-122
	RO (Reset Option)	5-123
	RP (Ready and Purge)	5-124
	RS (Remove Jobs from Schedule)	5-125
	RT (Remove Multipack File Table Entry)	5-127
	RY (Ready Peripheral)	5-128
	SB (Seconds Before decay)	5-129
	SD (Assign Additional System Drives)	5-130
	SE (Switch Enable)	5-131
	SECURITY	5-132
	SL (Set LOG)	5-133
	SM (Set Database Parameters)	5-134
	SN (Assign a Tape Serial Number)	5-137
	SNL (Assign a Tape Serial Number and Lock)	5-138
	SO (Set Option)	5-139
	SP (Change Schedule Priority)	5-140
	SQ (Squash Disk)	5-141
	ST (Suspend Processing)	5-143
	START	5-144
	SV (Save Peripheral Unit)	5-145
	SW (Set Switch)	5-146
	SY (System Program Pack)	5-147
	TD (Time and Date)	5-149
	TG (Trace Gismo)	5-150
	TI (Time Interrogation)	5-152
	TL (Transfer LOG)	5-153
	TO (Display Options)	5-154
	TR (Time Change)	5-155
	TS (Test Switches)	5-156
	UL (Assign Unlabeled File)	5-157
	USER	5-158

TABLE OF CONTENTS (Cont)

Section	Title	Page
	WD (Display MCP Date)	5-159
	WFL (Work Flow Language) Command	5-160
	WM (Display Current MCP and Interpreter)	5-161
	WS (Display Schedule)	5-162
	WT (Display MCP Time)	5-163
	WW (List Contents of the Name Table)	5-164
	WY (Mix and Status Interrogation)	5-167
	XC (Remove Segments Temporarily)	5-168
	XD (Remove Segments Permanently)	5-169
6	NETWORK CONTROLLER OPERATIONS	6-1
	Network Controller Execution	6-1
	Network Controller Priority	6-1
	Network Controller Program Switches	6-1
	Program Switch 2	6-1
	Program Switch 3	6-2
	Program Switch 7	6-2
	Network Controller Input Commands	6-3
	Common Syntax	6-3
	<line-id>	6-3
	<station-id>	6-3
	ODT Commands	6-5
	CLOSE Command	6-5
	CREATE Command	6-6
	DATARATE Command	6-7
	DUMP Command	6-8
	HELP Command	6-9
	IOLOG Command	6-10
	MAKE Command	6-11
	QUIT Command	6-12
	RELOAD LINE Command	6-13
	RELEASE LINE Command	6-14
	RESTART LINE Command	6-15
	STANDBY Command	6-16
	STATISTICS Command	6-17
	STATUS Command	6-18
	Data Communications ODT	6-20
	Differences Between Data Communication and ODT-Control ODT Device	6-20
	Using the Data Communications ODT as a Terminal	6-21
7	SYSTEM OUTPUT MESSAGES	7-1
	Output Message Syntax	7-1
	<task-specifier>	7-1
	<priority>	7-2
	<abort-reference>	7-2
	<time>	7-3
	<date>	7-3

TABLE OF CONTENTS (Cont)

Section	Title	Page
8	SYSTEM HALTS	8-1
	FIRMWARE HALTS (L = @00FOxx@ OR @000Fxx@)	8-2
	GISMO HALTS (L = @0D00xx@)	8-3
	MICRO MCP HALTS (L = @0200xx@)	8-5
	MCP HALTS (L = @000011@)	8-6
	Coldstart Tape Halts	8-16
A	MCP MEMORY MANAGEMENT	A-1
	Concepts and Definitions	A-1
	Memory Fragmentation	A-1
	Working Set	A-2
	Thrashing	A-2
	B 1000 Memory Management Algorithms	A-4
	Level One (First-In, First-Out)	A-4
	Level One Advantages	A-4
	Level One Disadvantages	A-4
	Level Two (First-In, First-Out With Thrashing Detection)	A-5
	Level Two Advantages	A-6
	Level Two Disadvantages	A-6
	Level Three (Memory Priority With Thrashing Detection)	A-6
	Level Three Advantages	A-7
	Level Three Disadvantages	A-7
	Extended Segment Decay	A-7
	Extended Segment Decay Advantages	A-8
	Extended Segment Decay Disadvantages	A-8
	Functional Characteristics	A-8
	Thrashing Detection	A-8
	Priority Memory Management	A-9
B	SYSTEM PERFORMANCE MONITORING	B-1
	System Console	B-1
	Operational Details	B-2
	Invoking The Performance Monitoring Capability	B-2
	MCP Performance Monitoring Options	B-2
	Specification Of Activities To Be Displayed	B-3
	Variable and Fixed Lamp Displays	B-7
	Bar Graph	B-7
	Fixed Lamp Display	B-9
C	DISK FILE ACCESS METHODS	C-1
	Logical Versus Physical I/O	C-1
	File Information Block	C-1
	I/O Descriptors and File Buffers	C-1
	Disk File Header	C-2
	Disk File Access Characteristics	C-2
	Serial Files	C-2
	Input Serial Files	C-2
	Output Serial Files	C-3
	Input/Output Serial Files	C-3

TABLE OF CONTENTS (Cont)

Section	Title	Page
	EXTEND Attribute Set	C-3
	EXTEND Attribute Reset	C-4
	Random Files	C-4
	Delayed Random Files	C-5
	AUDITED and PROTECTED Files	C-6
	AUDITED FILES	C-6
	PROTECTED FILES	C-6
	Disk File Access Specifications	C-7
	Source Program Specifications	C-7
	BASIC File Declarations	C-8
	COBOL File Declarations	C-8
	FORTRAN File Declarations	C-8
	SDL/UPL File Declarations	C-9
	RPG File Declarations	C-10
	FILE Attribute Specifications	C-11
D	FILE SECURITY	D-1
	USER Command	D-1
	File Security System	D-2
	File Security - Batch Processing	D-3
	Operation of File Security System	D-3
	Non-Privileged Usercode Handling	D-8
E	NOTATION CONVENTIONS	E-1
	Left and Right Brackets ([])	E-1
	Left and Right Broken Brackets (< >)	E-1
	At Sign (@)	E-1
	Railroad Diagrams	E-1
	Required Items	E-2
	Optional Items	E-2
	Loops	E-3
	Bridges	E-4
	INDEX	1

LIST OF ILLUSTRATIONS

Figure	Title	Page
1-1	Diagram of Task Schedules and System Mix	1-2
A-1	Thrashing Point	A-3
A-2	Thrashing Detection Logic Flowchart	A-10
A-3	Memory Sweeper Logic Flowchart.	A-11

LIST OF TABLES

Table	Title	Page
5-1	Utility Programs and their Associated System Command	5-147
B-1	Switch Specification Summary	B-4
B-2	I/O Activity Switch Specifications	B-4
B-3	Processor Activity Switch Specifications	B-5
B-4	Overlay Activity Switch Specifications	B-6
B-5	B 1000 Bar Graph Scale Factor	B-8
C-1	Disk File Access Specification Summary	C-7
C-2	RPG Disk File Access Technique Specifications	C-10
D-1	File Names Created by the CHECK/WRITER Program.	D-8
D-2	Actual File Names Used by the MCP	D-8
D-3	Actual File Names Used When Creating a New File	D-8
D-4	File Names Used When Accessing an Existing File	D-9

INTRODUCTION

PURPOSE OF MANUAL

The B 1000 Systems System Software Operation Guide is a reference manual for computer operators. Its purpose is to provide all the necessary details to operate the B 1000 system software.

ORGANIZATION OF MANUAL

This manual consists of seven sections and two appendices. Each is briefly described as follows:

Section	Contents
1	INTRODUCTION TO THE SYSTEM Provides a brief introduction to the B 1000 computer system and lists the related documents.
2	OVERVIEW OF CONTROL INSTRUCTIONS Provides an overview, by function, of the system control instructions.
3	INITIALIZING THE SYSTEM Provides the necessary information to initialize a B 1000 computer system.
4	PROGRAM CONTROL INSTRUCTIONS Provides a description of all the program control instructions available in the B 1000 operating system.
5	SYSTEM CONTROL INSTRUCTIONS Provides a description of all the system commands available in the B 1000 operating system.
6	NETWORK CONTROLLER OPERATIONS Provides a description of the network controller operations.
7	SYSTEM OUTPUT MESSAGES Provides an alphabetical list of all the B 1000 operating system output messages and the operator actions required.
8	SYSTEM HALTS Provides a list of the B 1000 operating system halts and the operator actions required.
A	MCP MEMORY MANAGEMENT Provides a functional description of memory management on the B 1000 computer system.

- B **SYSTEM PERFORMANCE MONITORING**
Provides the operational details of the system performance monitoring capabilities on the B 1000 computer system.
- C **DISK FILE ACCESS METHODS**
Provides a functional description of the disk file access methods available through the B 1000 operating system.
- D **FILE SECURITY**
Provides the operating details of the file security capabilities of the B 1000 operating system.
- E **NOTATION CONVENTIONS**
Describes the notation conventions used in this manual.

RELATED DOCUMENTATION

The following documents are referenced in this document:

B 1000 Systems System Software Operation Guide, Volume 2, form 1152097.

B 1000 Systems Network Controller Installation Manual, form 1152196.

B 1000 Systems Network Definition Language (NDL) Language Manual, form 1073715.

B 1000 Systems CANDE Installation and Operation Manual, form 1152006.

B 1000 Systems Supervisory Message Control System (SMCS) Installation, Operation, and Functional Description Manual, form 1108891.

B 1000 Systems COBOL74 Language Manual, form 1168622.

B 1000 Systems Report Program Generator II (RPGII) Language Manual, form 1152063.

B 1000 Systems Burroughs Network Architecture (BNA) Installation and Operation Manual, form 1151974.

B 1000 Systems Work Flow Language (WFL) Language Manual, form 1138039.

SECTION 1 INTRODUCTION TO THE B 1000 SYSTEM

THE MASTER CONTROL PROGRAM (MCP)

The Master Control Program (MCP) is a modular operating system that handles complex and repetitive functions and makes programming and operations efficient and productive. The MCP provides the best coordination and processing control for system throughput by allowing maximum use of all system components. Operator intervention is greatly reduced through complete resource management. Because all program functions are performed under this centralized control, changes in scheduling, system configuration, and program size can be readily accommodated, resulting in greater system throughput.

SYSTEM DESCRIPTION

The following functions are controlled by the MCP.

- Loading
- Handling interrupts
- Controlling input/output (I/O)
- Selecting and initiating programs
- Handling I/O errors
- Maintaining the system log
- Allocating memory and disk storage
- Overlaying data and code segments
- Multiprogramming

The MCP services the following peripheral equipment.

- 96-column card devices
- 80-column card devices
- Data Communications (single-line and multiline controls)
- Disk Cartridges
- Disk Packs
- Diskettes (Floppy Disks)
- Head-per-Track disks
- Line Printers
- Magnetic Tape Cassettes
- Magnetic Tapes (7-track, 9-track, and phase-encoded)
- MICR Reader-Sorters
- Operator Display Terminal (ODT)
- Paper Tape Devices

DEVICE REQUIREMENTS

The following equipment is required for the B 1000 operating system. This equipment is not dedicated to the operating system and can be used by any user-written program.

Device	Usage
ODT or Console Printer	Operator Communication
Disk	Auxiliary storage

CENTRAL SERVICE MODULE

The Central Service Module (CSM or GISMO) is a microcoded routine that performs the following functions in an equivalent hard-wired machine:

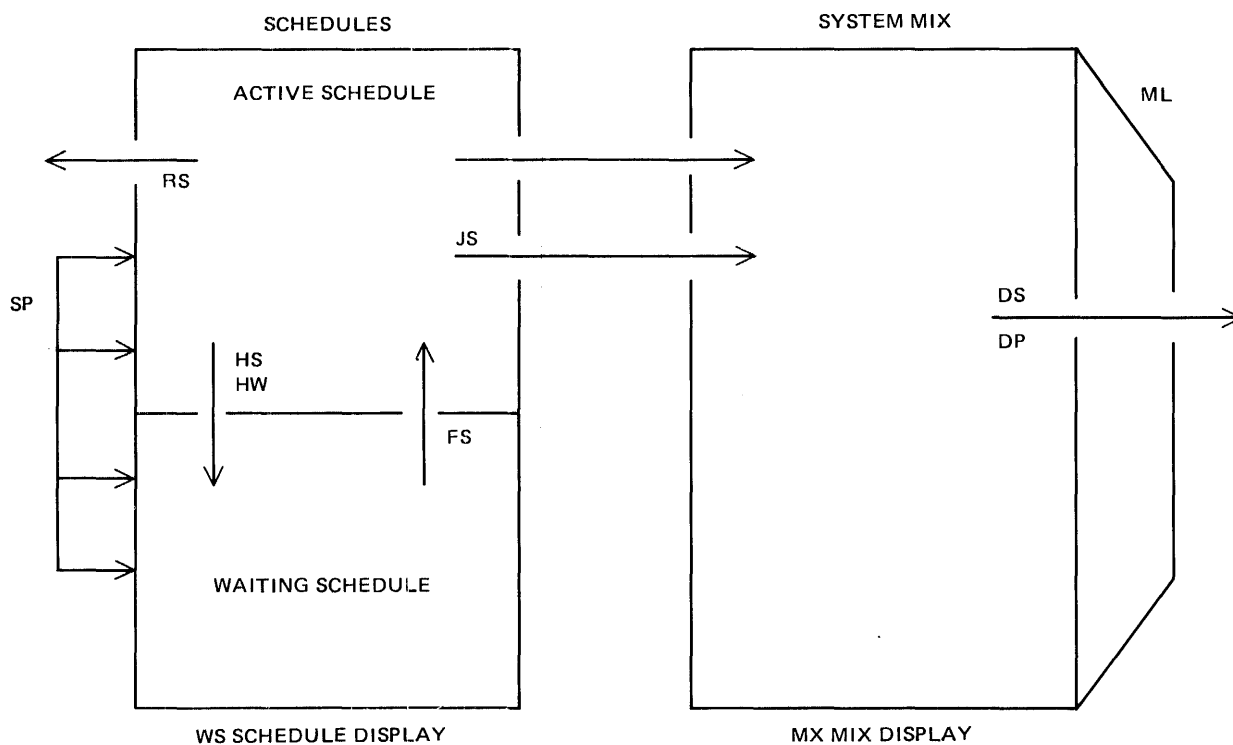
- Handling and detecting interrupts.
- Passing control to or from the MCP, usually on an interrupt.
- Controlling all I/O activity, such as I/O initialization, data transfers, and I/O termination.
- Managing interpreter activity.

INTERPRETERS

Interpreters are microcoded routines, or firmware, that perform the operations specified by the program. Each language has its own interpreter.

TASK SCHEDULES AND THE SYSTEM MIX

Figure 1-1 shows the overview of the task schedules and the system mix, and the commands that affect the movement of tasks from the waiting schedule, active schedule, and the system mix.



G18608

Figure 1-1. Diagram of Task Schedules and System Mix

The basic operation of the B 1000 operating system waiting schedule, active schedule, and mix are described in the following paragraphs.

Waiting Schedule

The waiting schedule contains the tasks that are currently waiting for the completion of another task or have been explicitly held in the waiting schedule by the HW or HS system commands. The following commands are allowed for tasks that are in the waiting schedule.

FS System Command

The FS system command causes the task to be removed from the waiting schedule and enter the active schedule.

HW System Command

The HW system command designates that certain tasks are to be placed in the waiting schedule to await the end of job of another task or until the system operator enters the FS system command.

RS System Command

The RS system command removes a specified task from the active or waiting schedule.

SP System Command

The SP system command changes the schedule priority of a task in the waiting schedule.

WS System Command

The WS system command lists the status of the tasks in the waiting and active schedules.

Active Schedule

The active schedule contains tasks that are candidates to enter the system mix. The task moves into the system mix when all its required resources are available. The following system commands are allowed for tasks in the active schedule.

HS System Command

The HS system command causes the specified task to be placed in the waiting schedule.

JS System Command

The JS system command causes the operating system to check the active schedule and attempt to move the top entry into the mix if memory is available.

RS System Command

The RS system command removes a specified task from the active or waiting schedule.

SP System Command

The SP system command changes the schedule priority of a task in the active or waiting schedule.

WS System Command

The WS system command lists the status of the tasks in the waiting or active schedule.

System Mix

The following system commands affect the status of tasks in the system mix.

DP System Command

The DP system command dumps a task's memory and discontinues the task system mix.

DS System Command

The DS system command discontinues a task in the system mix.

ML System Command

The ML system command sets the maximum number of jobs that can be in the system mix below the priority of 9.

MX System Command

The MX system command displays the current tasks in the system mix.

CONTROL INSTRUCTIONS

The MCP is directed to perform particular actions by the system operator through the use of control instructions.

There are two types of control instructions: program control instructions (described in section 4) and system commands (described in section 5).

The following rules apply to all control instructions supplied to the MCP.

- If the percent sign (%) character appears in a control instruction, all information following the percent sign (%) character is ignored for control purposes. This allows comments to be included with the control instructions for documentation purposes. However, if the percent sign (%) character is enclosed within double quotation mark (") characters, the percent sign (%) character is used as part of the control instruction and is not treated as a comment.
- Any program name or file name containing the following special characters must be enclosed within a double quotation mark (") character.

Character Symbol	Character Name
;	semicolon
,	comma
=	equal sign
/	virgule
"	double quotation mark
@	at sign
%	percent sign
a - z	lower-case letters

Any special characters not contained in this list do not require a double quotation mark (") character to enclose the name.

Example 1:

```
"FILE%001"
```

Example 2:

```
"%03"/"%0ABC ="
```

The virgule (/) character in this example separates the first name from the file name and is not enclosed within double quotation mark (") characters.

Example 3:

```
"/XYZ"
```

The virgule (/) character is part of the first name and must be enclosed within double quotation mark (") characters.

Example 4:

```
TESTFILE/#0000000001
```

The pound sign (#) character is not listed as a special character and does not need to be enclosed within double quotation mark (") characters.

All control instructions are described in the following paragraphs under headings that imply that each instruction must consist of a separate card image. This is not necessary. If the text of one control instruction is delimited by a space or a semicolon character, this is considered the logical end of that control instruction.

SOURCES OF CONTROL INSTRUCTIONS

The following paragraphs describe three sources of control instructions: 1) punched cards, 2) the Operator Display Terminal (ODT), and 3) the ZIP statement.

Punched Cards

If punched cards are used to communicate a control instruction to the MCP, the following rules apply.

Column 1 of the first control card must contain an invalid character for 80-column cards or a question mark (?) character for 96-column cards. An invalid character or question mark (?) character cannot appear in any other column. Columns 2 through 72 of the card can contain control instructions in free-field format. Control information is limited to the first 72 columns of the card.

Operator Display Terminal (ODT)

Control instructions can be communicated to the MCP from the operator display terminal (ODT) by the four steps that follow.

1. Place the unit in LOCAL mode.
2. Set the cursor to the HOME (upper, left-hand corner) position. If the control instruction does not fit within the space provided at the top of the screen, use the KB INP.LINES system command to increase the reserved space as necessary.
3. Enter the control instruction. Errors can be corrected directly by using the cursor-positioning keys.
4. Press the XMT key and the control instruction is read by the MCP.

When the MCP has read the control instruction, the ODT is left in receive (RCV) mode to allow displaying of any responses by the MCP. If it is necessary to view the screen for any length of time, place the ODT in LOCAL mode so the MCP is not able to change the screen. The operator must leave the ODT in RCV mode when not entering a control instruction.

It is often necessary to view messages that are no longer displayed on the screen. Scrolling the display backwards or forwards is possible by using the KB system command.

ZIP Statement

Control instructions can be communicated to the MCP by the use of a ZIP statement in an executing program. The ZIP statement in the program must reference a defined data area where the control statement is located. Refer to the appropriate language reference manual for the specific syntax regarding the ZIP statement.

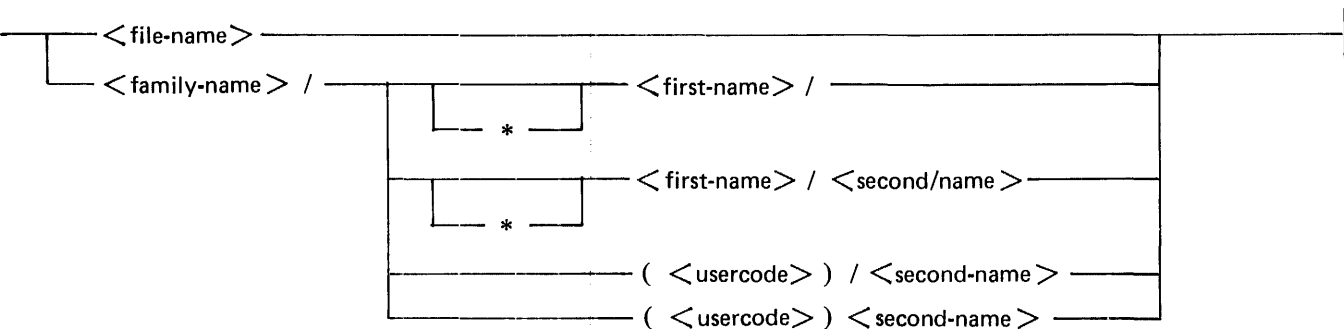
GENERIC TERMS

A number of generic terms are used within this manual to describe the syntax of system commands and output messages. These terms are defined in the following paragraphs.

file-identifier

A file-identifier term identifies and references a file.

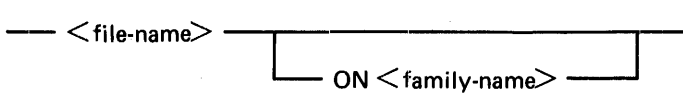
Syntax:



file-title

A file-title term references a file and the name of the disk on which it resides.

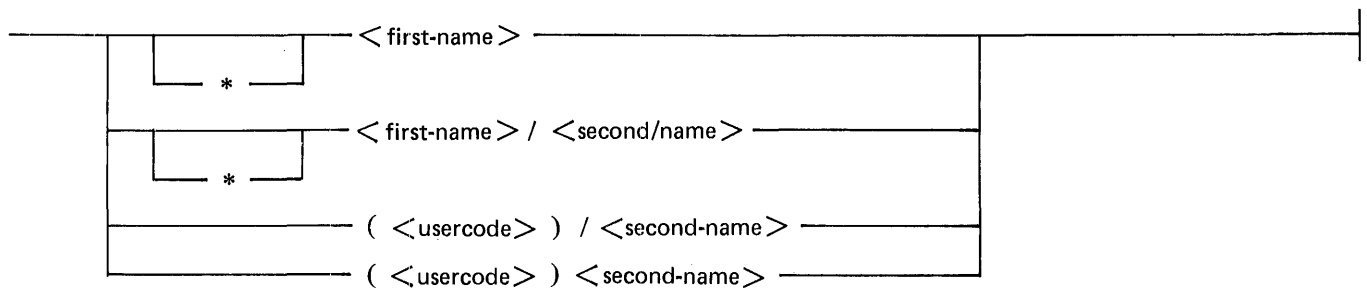
Syntax:



file-name

A file-name term references a file without regard to the media on which it resides.

Syntax:



Semantics:

<family-name>

A <family-name> is the name of a disk or a pack. A <family-name> can also be the name of a tape volume in some Work Flow Language statements. A <family-name> consists of one to ten characters. A <family-name> of DISK refers to the system disk.

*

The asterisk character is used in a file name to specify that the usercode is NOT to be applied to the file name.

<first-name>

A <first-name> is the first part of a <file-name>. A <first-name> consists of one to ten characters. When the asterisk syntax is used, the asterisk character counts as one of the ten characters in the <first-name>.

<second-name>

A <second-name> is the second part of a <file-name>. A <second-name> consists of one to ten characters.

<usercode>

A <usercode> is a name assigned to a user to secure system and file access. When a <usercode> in parentheses is the first part of a <file-name>, the file belongs to the user represented by <usercode>. A <usercode> consists of one to eight characters.

program-name

The program-name term is a file-identifier which is the name of a program.

compiler-name

The compiler-name term is a file-identifier which is the name of a B 1000 compiler.

interpreter-name

The interpreter-name term is a file-identifier which is the name of a B 1000 interpreter.

unit-mnemonic

The unit-mnemonic term is a name that consists of one to three characters and identifies a peripheral device.

The following unit-mnemonic terms are assigned to the peripherals attached to the system by the operation system.

Unit-Mnemonic Name	Device
CDx	Card Reader/Punch
CPx	Card Punch
CRx	Card Reader
CSx	Magnetic Tape Cassette
DCx	Disk Cartridge
DKx	Head-per-Track Disk
DPx	Disk Pack
FDx	Diskette ("Floppy Disk")
LPx	Line Printer
MTx	Magnetic Tape
ODT	Operator Display Terminal
PPx	Paper Tape Punch
PRx	Paper Tape Reader
SRx	MICR Reader-Sorter

The letter x represents any one of the upper-case letters A through Z, or digits 1 through 9, for assignment to multiple units of a specified type. The units begin with the upper-case letter A through the upper-case letter Z, then continue with the digits 1 through 6.

SYSTEM disk

A SYSTEM disk is a disk pack that is initialized as a SYSTEM type pack. Refer to the PACK/INIT or SYSTEM/DISK-INIT utility programs for a complete description of how to initialize a disk pack as a SYSTEM disk. One or more SYSTEM disks must be present on the system for the MCP to function. Head-per-track disk is always considered a SYSTEM disk. The <pack-id> or <family-name> of a SYSTEM disk is always DISK. The name of the actual disk, which is displayed in response to an OL system command may be anything desired.

user disk

A user disk is a disk pack or cartridge initialized as a USER type pack. The MCP does not need user disks present to function. No user disk may be named DISK.

mix number

The mix number is a number assigned to a program by the operating system when it enters the active schedule. This number must be specified when the operator desires to change the state of an executing program or give messages to a program.

SECTION 2

OVERVIEW OF CONTROL INSTRUCTIONS

This section contains an overview of the control instructions available to the operator of the B 1000 system. There are two types of control instructions: program control instructions and system commands. Program control instructions are described in detail and are listed in alphabetical order in section 4. System commands are described in detail and are listed in alphabetical order in section 5.

Program control instructions and system commands are grouped into functional categories in this section. A brief description of each instruction is included. This section is intended as a cross reference for an operator who needs to perform a particular function and needs to see a group of control instructions related to that function.

Some control instructions can be used to perform more than one function and, therefore, appear in more than one functional category.

INTERROGATING AND CHANGING THE TIME AND DATE

DR (Date Reset)

Changes the current date maintained by the MCP.

TD (Time and Date)

Displays the current time and date maintained by the MCP.

TR (Time Reset)

Changes the current time maintained by the MCP.

WD (What Date)

Displays the current date maintained by the MCP.

WT (What Time)

Displays the current time maintained by the MCP.

INTERROGATING THE STATUS OF PROGRAMS IN THE MIX

CU (Core Usage)

Displays the amount of system memory in use by one or more executing programs.

DB (Data Base Status)

Displays the active DMSII data bases.

MX (Mix)

Displays the priority and status of one or more programs in the mix.

QP (Query Program)

Displays the control attributes of an executing program.

TI (Time In)

Displays the amount of processor time accumulated for an executing program.

TS (Test Switches)

Displays the switch settings of an executing program.

WM (What MCP)

Displays which MCP and related system software are currently being used.

WY (WHY)

Displays the priority and status of one or more programs in the mix.

RESPONDING TO PROGRAM MESSAGES

AC (Accept)

Precedes a response to an ACCEPT message from a program; cannot be delimited with a semicolon (;) character.

AX (Accept)

Precedes a response to an ACCEPT message from a program; can be delimited with a semicolon (;) character.

FM (Forms Mounted)

Response to a SPECIAL FORMS REQUIRED message from a program.

OF (Optional File)

Response to a NO FILE message from a program when the file has the OPTIONAL file attribute specified.

STATUS CHECKING

AB (Auto Backup)

Displays the current autoprint value or reserves a line printer for the autoprint function.

BB (Backup Blocks)

Displays or sets the number of blocks per area assigned to printer backup files.

BD (Backup Disk)

Displays or sets the disk name for printer backup files.

BF (Backup Files)

Displays all current printer and punch backup files.

CD (Card Decks)

Lists all pseudo decks on disk.

CU (Core Usage)

Interrogates the amount of memory in use by any or all programs running.

DB (Data Base)

Determines which DMSII data bases are in use.

DF (Date of File)

Displays the creation date of a disk file.

DL (Disk Location)

The DL system command designates the system default pack for backup files and dump files.

FN (File Name)

Displays the internal and external file names currently in effect in a program.

- HN (Host Name)
Displays the logical hostname of the system.
- IC (Interpreter Count)
Displays the current size of the MCP interpreter dictionary.
- ML (Mix Limit)
Displays the current MCP setting for the number of jobs with a priority less than nine allowed in the mix.
- MP (Memory Priority) *MM*
Displays the memory priority of a running program.
- MU (Multipack File)
Displays the multipack file information table.
- MX (Mix)
Displays the status of all tasks in the mix.
- NET (BNA Network Mode Inquiry)
Displays the current network mode of the B 1000 BNA system.
- OL (Output Label)
Displays the label of the device specified in the request.
- PD (Print Directory)
Displays whether or not a file is in the disk directory.
- PF (Print Fetch)
The PF system command displays messages from a task that contains a Work Flow Language (WFL) FETCH specification and is awaiting a fetch action.
- PP (Processor Priority)
Displays the processor priority of a running program.
- PR (Priority) *PV*
Displays the processor and memory priorities of a running program.
- QF (Query File)
Displays the program control attributes of a program on disk.
- QP (Query Program)
Displays the program control attributes of a running program.
- SM (Set DB Parameters)
Displays the current DMSII data base parameters.
- TD (Time and Date)
Displays the current time and date.
- TI (Time Interrogate)
Displays the current processor time used by a running program.
- TO (Type Options)
Displays the current setting of the operating system options.

TS (Test Switches)

Displays the current switch settings of a task in mix.

WD (What Date)

Displays the current date.

WM (What MCP)

Displays the name of the executing version of the operating system.

WS (What Schedule)

Displays the contents of the waiting and active schedules.

WT (What Time)

Displays the current time.

WW (Name Table)

Displays the entries in the Name Table.

WY (Why)

Displays the status of all tasks in the mix.

MESSAGES CONCERNING PROGRAM STATUS

DS (Discontinue)

Terminates a program and closes all files.

DP (Dump and DS)

Initiates a program memory dump and discontinues the task.

IL (Ignore Label)

Ignores a label on an input device when FILE NOT PRESENT condition is encountered for a program that requires the input device.

MR (Most Recent File)

Discards the most recent file and saves the old file when a DUPLICATE FILE ON DISK condition occurs.

OK (OK)

Continues a program that was suspended by either of the following conditions.

1. The program was suspended by the ST command.
2. The program was suspended waiting for a disk file, more disk space, or more memory and the file, disk space, or memory, respectively, was made available.

OU (Output Unit)

Directs an output file to the device specified.

RM (Remove)

Removes the old copy of the file and replaces it with the new copy when a DUPLICATE FILE ON DISK condition occurs.

UL (Unlabeled File)

Assigns an unlabeled tape file to a program from a specified unit. The unit must be ready.

DISK OPERATION AND INTERROGATE MESSAGES

CW

- DF (Date of File)
Displays the creation date of a file.
- FN (File Name)
Displays the internal and external file names of the program.
- IL (Ignore Label)
Assigns a different input device for an input file.
- KA (Analyze Disk)
Prints disk file information on the line printer.
- KC (Print Disk Segments in Character Format)
Prints designated disk segments on the line printer in character format.
- KP (Print Packed Disk Segment)
Prints designated disk segments on the line printer in hexadecimal format.
- MH (Modify Header)
Changes the attributes of a disk file.
- MU (Multipack File)
Displays the multipack file information table.
- OL (Output Label)
Displays the current label of a device.
- PD (Print Directory)
Interrogates whether or not a file is on disk.
- PF (Print Fetch)
The PF system command displays messages from a task that contains a Work Flow Language (WFL) FETCH specification and is awaiting a fetch action.
- PG (Purge)
Purges the disk directory of a designated disk and makes the disk a scratch disk ready for output.
- PO (Power Off)
Makes a phase encoded (PE) tape or disk device go offline. This command should be used prior to removing tape or disk media from a device.
- PV (Pack Override)
Displays, sets, or resets the pack override bit for a usercode.
- QF (Query File)
Displays the control attributes of a program.
- RC (Recover Data Base)
Recovers a DMSII data base when the data base needs recovery.
- RL (Relabel)
Changes the label on a user disk.

RE

RT (Remove Multipack File Table Entry)
Removes the multipack file information table.

SM (Set Data Base Parameters)
Sets the DMSII parameters of the data base.

SQ (Squash)
Reallocates disk space to reduce checkerboarding.

07

XC (Lockout Disk)
Temporarily removes disk segments until the next clear/start operation.

XD (Lockout Disk)
Permanently removes disk segments until the disk can be initialized.

HANDLING THE SYSTEM OPTIONS

IC (Interpreter Count)
Sets the number of entries allowed in the interpreter dictionary of the operating system (MCP).

KB (Keyboard Options)
Interrogates or modifies the message formatting characteristics of the Operator Display Terminal.

50

RO (Reset Option)
Resets the designated operating system option.

MA

SL (Set Log)
Invokes the logging functions of the operating system.

SO (Set Option)
Sets the designated operating system option.

TO (Type Options)
Displays the current settings of the operating system options.

HANDLING MEMORY DUMPS

DM (Dump Memory and Continue)
Dumps memory assigned to the designated program and then continues execution.

DP (Dump and Discontinue)
Dumps memory assigned to the designated program and then discontinues the program.

PM (Print Memory Dump)
Analyzes and prints the designated memory dump. If used without a designated dumpfile number, the system memory dump is processed.

HANDLING MAGNETIC TAPE DEVICES

CL (Clear Unit)
Clears the unit. The user program is discontinued.

FR (Final Reel)
Notifies the operating system (MCP) of the last reel of an unlabeled tape file.

- IL (Ignore Label)
Ignores the label on a designated tape drive and accepts the current tape as input.
- OL (Output Label)
Displays the label name of a tape on the designated tape unit.
- PG (Purge)
Purges the label on a designated tape unit and makes the tape a scratch label ready for output.
- PO (Power Off)
Unloads a phase encoded (PE) magnetic tape.
- RY (Ready)
Readies a tape that has been locked.
- RP (Ready and Purge)
Readies a locked tape and purges it to create an output scratch tape.
- SN (Serial Number)
Initializes a new tape with the new specified serial number.
- UL (Unlabeled)
Accepts an unlabeled tape into the system after a FILE NOT PRESENT condition occurs. Tape unit must be ready.

HANDLING PERIPHERALS

- AL (Align Forms)
Prints one line on the line printer to assist the operator in lining up a special form.
- CL (Clear Unit)
Discontinues a program using the designated unit and readies the unit for subsequent use.
- CQ (Clear the Queue)
Clears messages backed up in the ODT queue.
- KB (Keyboard Options)
Controls ODT and may redesignate the line printer for output or recall ODT messages.
- LT (Load Train Printer Table)
Loads appropriate train printer translation table under control of the operating system (MCP) if changing train.
- OL (Output Label)
Displays the label name on the designated unit if in use by a program or status (ready or not).
- OU (Output Unit)
Directs output to the designated unit.
- RY (Ready)
Readies a peripheral device for the operating system (MCP).
- SV (Save)
Makes a peripheral not ready to the operating system (MCP) until an RY system command is entered to change the peripheral status to READY.

CHANGING THE STATUS OF PROGRAMS RUNNING IN MIX

- DP (Dump and Discontinue)
Dump memory and discontinue program execution.
- DS (Discontinue)
Discontinues the designated program.
- GO (Go)
Resumes a program that has been suspended.
- LP (Lock Program)
Locks a program in the mix so it cannot be accidentally discontinued by the operator.
- ML (Mix Limit)
Changes the upper limit to the number of jobs able to run in the mix.
- MP (Memory Priority)
Changes the memory priority of a running task.
- NC (Network Controller Command)
Sends the associated message to the network controller.
- PP (Processor Priority)
Changes the processor priority of a running task.
- PR (Priority Change)
Changes the processor priority of a running task.
- MM (Memory Management)
Controls certain attributes of the memory management system.
- SB (Seconds Before Decay)
Sets the number of seconds that a code segment in memory has before the memory priority of the code segment is decremented by 1.
- ST (Stop)
Suspends a program in the mix and rolls it out to disk.
- SW (Set Switches)
Sets the switches of a program in the mix as designated.
- Handwritten notes:*
MIX
LP 20
MCS

PERFORMING HEXADECIMAL CONVERSION AND CALCULATIONS

- CP (Compute)
Computes binary, hexadecimal, and octal conversion as well as basic arithmetic calculations.

CONTROLLING PROGRAMS IN THE SCHEDULE

- FS (Force from Schedule)
Forces the designated program from the waiting schedule to the active schedule.

HS (Hold in Schedule)

Moves a program from the active schedule to the waiting schedule and holds it until an FS command is entered for the program.

HW (Hold in Waiting Schedule)

Holds a program in the waiting schedule until a designated program goes to end of job.

JS (Jiggle Schedule)

Causes the MCP to re-check the active schedule for jobs to move into the mix.

RS (Remove from Schedule)

Removes the designated task number from the schedule.

SP (Schedule Priority)

Changes the schedule priority to a new level. If there is a task hung in the active schedule (for example, waiting for an interpreter) and other tasks are waiting for that task to move into the mix, then the SP command can be used to lower the schedule priority of the hung task so that the other tasks can enter the mix.

WS (What Schedule)

Displays the contents of the active and waiting schedules.

CONTROLLING BACKUP (SPOOLING) FUNCTIONS

AB (Automatic Backup)

Designates a line printer to automatically print backup files as they are released.

AL (Align Forms)

Prints one line on the line printer to assist the operator in lining up a special form.

AP (Autoprint)

Inserts a printer backup file into the autoprint queue.

BB (Backup Blocks Per Area)

Displays or sets the number of blocks per area to be used when creating printer backup files.

BD (Backup Disk)

Displays or assigns the user disk for backup files.

BF (Backup Files)

Displays the backup files and dumpfiles currently on disk.

DL (Disk Location)

The DL system command designates the system default pack for backup files and dump files.

OU (Output Unit)

Assigns a disk or tape device for a backup file in response to a PRINTER OR PUNCH REQUIRED condition.

PB (Print Backup)

Prints or punches designated backup files.

RB (Remove Backup Files)
Removes designated backup files (same as RF).

RF (Remove Backup Files)
Removes designated backup files (same as RB).

CONTROL OF PSEUDO CARD READERS

CD (Card Decks)
Displays the current pseudo decks on disk.

ED (Eliminate Decks)
Removes pseudo decks currently being read by pseudo readers.

IL (Ignore Label)
"#" designates a numbered pseudo reader.

LD (Load Control)
Initiates the system load control function.

RD (Remove Decks)
Removes pseudo decks on disk that are not currently in pseudo readers.

RN (Run Pseudo Readers)
Changes the number of pseudo readers.

HANDLING THE LOG FILES

EM (Engineers Message)
Enters a message into the engineers log.

ER (Error Rate)
Displays error rates on selected disk drive units or packs and optionally resets their historic error rate.

ET (Engineers Log Transfer)
Creates a new log and prints the old log in full or a log summary.

LC (Log Comment)
Enters a message into the system log file.

LN or LG (Logout)
Transfers and prints the system or ODT logs. Old logs can be removed when no longer needed. A new log is created after the transfer operation. Sends the associated message to the network controller.

SL (Set Log)
Initializes the system or ODT log file and sets the number of segments per area. Entering SL 0 resets the log file.

TL (Transfer Log)
Transfers the old system or ODT log file and creates a new system or ODT log without printing the contents of the old log.

CHANGING THE SYSTEM SOFTWARE ENVIRONMENT

CM (Change System Software)

Enters new software in the Name Table or purges an entry from the Name Table.

SD (System Disk)

Designates the disk units to be used as multipack extensions of the SYSTEM disk.

SE (Enable Console Switches)

Enables sensing of console switches by software while the system is running.

BURROUGHS NETWORK ARCHITECTURE (BNA) COMMANDS

HN (Host Name)

Displays or sets the logical host name of the B 1000 system.

NET (Network Mode Change or Inquiry)

Changes or queries the network mode of the B 1000 BNA system.

NW (Network Message)

Sends the text of a message to the BNA network services manager program.

JOBSTART (Initiate Job Transfer to Another Host)

Requests that the task file on disk be sent to the host in a BNA network and that the task be run there.

WORK FLOW LANGUAGE (WFL) COMMAND

START (Start a WFL Job)

Executes the WFL job task.

SECTION 3 INITIALIZING THE SYSTEM

HARDWARE INITIALIZATION

The following paragraphs describe the creation of system cassettes and user disks.

How to Create the System Cassettes

There are six system cassettes used in conjunction with the B 1000 operating system. They are as follows:

Cassette Name	Description
clear/start	Brings system to an operable state.
COLDSTART/DISK	Coldstarts system from a disk pack.
COLDSTART/TAPE	Coldstarts system from a SYSTEM tape.
DISK/DUMP	Dumps a disk pack sector-by-sector.
PACK/INIT	Initializes disks.
STANDALONE/DISK-DUMP	Dumps a disk pack file-by-file.

Creating the clear/start Cassette

To create a clear/start cassette, execute the SSLOAD/MAKCAS program with program switch 2 equal to 0 and enter the file identifier clear/start through the ODT with an AX (accept) system command. One clear/start cassette is created for each AX system command; however, the system operator must ensure that a scratch cassette (SN system command) is made available to the program, and the cassette drive is ready. A blank AX system command terminates the SSLOAD/MAKCAS program.

Note

This program uses the magnetic tape cassette I/O control, NOT the console cassette which can only read cassettes.

Example:

Operator Enters:

```
EXECUTE SSLOAD/MAKCAS;SWITCH 0 0;
```

ODT Output:

```
SSLOAD/MAKCAS = 1257 BOJ...
% SSLOAD/MAKCAS = 1257 ENTER FILE IDENTIFIER
SSLOAD/MAKCAS = 1257 ACCEPT.
```

Operator Enters:

```
1257AXclear/start
```

```
1257AX
```

ODT Output:

```
SSLOAD/MAKCAS = 1257 EOJ.
```

Stand-alone Utility Cassettes (SDL Utility Programs)

To create the stand-alone utility cassettes COLDSTART/DISK, COLDSTART/TAPE, DISK/DUMP, PACK/INIT, and STANDALONE/DISK-DUMP, execute the SSLOAD/MAKCAS program with program switch 0 equal to 1 and identify the input specification file. The input file identifies all software (including the loader, GISMO routine, interpreter, and utility programs) to be loaded onto the cassette. The input file must have the information in the exact format in the following description, one specification per record, each beginning in column 1.

```
L CASSETTE/LOADER  
G GISMO/SA  
I SDL/INTERP1U  
S <file identifier>
```

The CASSETTE/LOADER, GISMO/SA, and SDL/INTERP1U files are the standard system software, and these program names must not be changed. The <file identifier> specifies the name of the SDL utility program to be loaded onto the cassette. All programs to be loaded must exist on disk. Only one SDL utility program can be written to each cassette. The names of each of the SDL utility programs are as follows:

```
PACK/INIT  
DISK/DUMP  
STANDALONE/DISK-DUMP  
COLDSTART/DISK  
COLDSTART/TAPE
```

Example:

The following code can be punched on specification cards and used to execute the SSLOAD/MAKCAS program.

```
? EXECUTE SSLOAD/MAKCAS SW 0 1;  
? DATA CARDS  
L CASSETTE/LOADER  
G GISMO/SA  
I SDL/INTERP1U  
S DISK/DUMP  
? END
```

Refer to the SSLOAD/MAKCAS program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the creation of system cassettes.

How to Initialize a Disk Pack

A SYSTEM pack and multiple user disks must be initialized prior to being used on the B 1000 system. There are two system utility programs available for this initialization. They are the SYSTEM/DISK-INIT program and the PACK/INIT cassette. The SYSTEM/DISK-INIT program allows the operator to initialize a disk under the B 1000 operating system. The PACK/INIT cassette allows the system operator to initialize a disk without the B 1000 operating system running. If installing a new B 1000 computer system and a SYSTEM disk is not available, the PACK/INIT cassette must be used. Refer to the PACK/INIT or SYSTEM/DISK-INIT sections in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the PACK/INIT cassette and SYSTEM/DISK-INIT program, respectively.

HOW TO INITIALIZE THE OPERATING SYSTEM

The Master Control Program (MCP) is designed as an integral part of the system and serves a wide range of installations and users. Provisions are incorporated in the system to adapt the operation of the MCP to the particular requirements of a variety of installations. This is accomplished by incorporating different environments within the MCP that can be specified at the time of system initialization. Some of the environment options can be changed or set after the system is initialized by using an Operator Display Terminal (ODT).

To place the operating system in control of the system, the operating system must be loaded onto the system disk with the system's environment defined and the disk directory established. The SDL2 interpreter is loaded next to interpret the SMCP S-language. When this procedure is complete, the SDL2 interpreter reads, interprets, and executes the instructions of the SMCP.

The following are the three separate procedures performed during system initialization, thereby making the system operable.

1. The SYSTEM and user disks must be initialized.
2. A coldstart operation must be performed.
3. A clear/start operation must be performed.

How to Coldstart the Operating System

To coldstart the B 1000 operating system requires the following:

COLDSTART/DISK or COLDSTART/TAPE Cassette B 1000 System Software on a disk or SYSTEM tape

The B 1000 operating system can be coldstarted from another disk or SYSTEM tape containing the B 1000 system software.

Using the COLDSTART/TAPE Cassette

The following instructions are used to coldstart the B 1000 operating system from the SYSTEM tape using the COLDSTART/TAPE cassette.

1. Mount the SYSTEM tape on a magnetic tape device.
2. If the processor is not already halted (that is, the RUN light on the console is out), bring the system to an orderly halt using either the INTRPT or the INTERRUPT pushbutton on the system console. If the INTERRUPT pushbutton fails to halt the processor (for example, due to an uninterruptible software loop), press the HALT pushbutton. If the HALT pushbutton also fails to halt the processor, press the HALT and CLEAR pushbuttons simultaneously.
3. Place the COLDSTART/TAPE cassette in the console cassette tape drive, and ensure that the cassette rewinds to beginning of tape (BOT).
4. For B 1990 systems, enter MTR GO in the command display and press the TRANSMIT key.

For all other B 1000 systems:

- 1) Set the MODE pushbutton to the TAPE position.
- 2) Press the START pushbutton.
- 3) The cassette reads the bootstrap loader and the processor halts with the RUN light out. The L register must contain @AAAAAA@ at this time; if not, the cassette must be rewound and the procedure restarted at step 1).
- 4) Set the MODE pushbutton to the RUN position.
- 5) Press the START pushbutton.

5. Reading of the cassette continues, loading the COLDSTART/TAPE program into memory. When the entire program has been loaded, control is given to the COLDSTART/TAPE program to begin the coldstart procedure.
6. The coldstart operation is completed successfully when the system console lights display @000011@ in the L register and @AAAAAA@ in the T register. The B 1000 operating system can then be clear/started.

If the coldstart operation does not complete successfully, refer to the COLDSTART/TAPE section in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the action required to complete the coldstart operation.

Using the COLDSTART/DISK Cassette

The following instructions are used to coldstart the B 1000 operating system from a user disk using the COLDSTART/DISK cassette.

1. Mount the new SYSTEM disk to be coldstarted in the SYSTEM disk drive.
2. Mount the disk that contains the B 1000 system software.
3. If the processor is not already halted (that is, the RUN light on the console is out), bring the system to an orderly halt using either the INTRPT or the INTERRUPT pushbutton on the system console. If the INTERRUPT pushbutton fails to halt the processor (for example, due to an uninterruptible software loop), press the HALT pushbutton. If the HALT pushbutton also fails to halt the processor, press the HALT and CLEAR pushbuttons simultaneously.
4. Place the COLDSTART/DISK cassette in the console cassette tape drive, and ensure that the cassette rewinds to beginning of tape (BOT).
5. For B 1990 systems, enter MTR GO in the command display and press the TRANSMIT key.

For all other B 1000 systems:

- 1) Set the MODE pushbutton to the TAPE position.
 - 2) Press the START pushbutton.
 - 3) The cassette reads the bootstrap loader, and the processor halts with the RUN light out. The L register must contain @AAAAAA@ at this time; if not, the cassette must be rewound and the procedure restarted at step 1).
 - 4) Set the MODE pushbutton to the RUN position.
 - 5) Press the START pushbutton.
6. Reading of the cassette continues, loading the COLDSTART/DISK program into memory. When the entire program has been loaded, control is given to the COLDSTART/DISK program to begin the coldstart procedure.
 7. If the COLDSTART/DISK cassette is successfully loaded, the following prompt sequence appears on the ODT.

COLDSTART/DISK MARK <mark-level>.<patch-level><date + compile time>

ENTER OUTPUT DRIVE - <DCA, DPA OR DKA>

A valid response causes the next message to be displayed.

ENTER INPUT DRIVE - <DC?, DP? OR DKA>

A valid response causes the next message to be displayed.

IS COMPLETE COPY DESIRED? <YES OR NO>

If YES is entered, all the files on the input disk are copied to the newly-coldstarted system disk. These files are copied after the system software is loaded on the new disk. If NO is entered, only the required system software is loaded on the newly-coldstarted system disk.

IS DATA COMPARISON DESIRED? <YES OR NO>

A YES response causes the COLDSTART/DISK program to verify that all data is copied correctly to the newly-coldstarted system disk. An advisory message is printed for any file in which errors are found.

After the coldstart operation and optional complete copy are complete, the following message is displayed.

COLDSTART COMPLETE - CLEAR/START REQUIRED

The following is an example of this sequence.

ODT Output:

COLDSTART/DISK - 12.0.000(03/12/84 15:10)
ENTER OUTPUT DRIVE - <DPA OR DKA>

Operator Enters:

DKA

ODT Output:

ENTER INPUT DRIVE - <DP? OR DKA>

Operator Enters:

DPB

ODT Output:

IS COMPLETE COPY DESIRED? <YES OR NO>

Operator Enters:

YES

ODT Output:

COLDSTART COMPLETE - CLEAR/START REQUIRED

If the coldstart operation is not successfully completed, refer to the COLDSTART/DISK section in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the COLDSTART/DISK cassette operation.

How to clear/start the Operating System

The following procedure must be used to perform a clear/start operation:

1. If the processor is not already halted (that is, the RUN light on the console is out), bring the system to an orderly halt using either the INTRPT or the INTERRUPT pushbutton on the system console. If the INTERRUPT pushbutton fails to halt the processor (for example, due to an uninterruptible software loop), press the HALT pushbutton. If the HALT pushbutton also fails to halt the processor, press the HALT and CLEAR pushbuttons simultaneously.
2. Place the clear/start cassette in the console cassette tape drive, and ensure that the cassette rewinds to beginning of tape (BOT).
3. For B 1990 systems, enter MTR GO in the command display and press the TRANSMIT key.

For all other B 1000 systems:

- 1) Press the MODE pushbutton to obtain MTR.
 - 2) Press the CLEAR pushbutton, then press START.
 - 3) The cassette reads the bootstrap loader, and the processor halts with the RUN light out. The L register must contain @AAAAAA@ at this time; if not, the cassette must be rewound and the procedure restarted at step 1).
 - 4) Any temporary environment changes to be made (such as a memory dump request) must be entered in the appropriate registers at this time. Refer to Temporary Operating Environment Changes within this section for details.
 - 5) Press the MODE pushbutton to obtain NORMAL.
 - 6) Press the START pushbutton.
4. Reading of the cassette continues, loading the clear/start program into memory. When the entire program has been loaded, control is given to the clear/start program to begin the system initialization procedure. The cassette is then rewound by pressing the REWIND pushbutton. The rewind is done automatically on B 1900 systems.

If the clear/start operation does not complete successfully, refer to the clear/start section in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the operation of the clear/start cassette.

Temporary Operating Environment Changes

Several changes to the default selections made by the clear/start procedure can be specified by entering parameters in certain registers during the clear/start process. The change requests are made by entering information through the hardware toggles and switches. The operator can make changes by doing the following.

1. Set the REGISTER SELECT and GROUP SELECT dials to point to the appropriate register.
2. Flip one or more of the 24 toggle switches upward to represent those bits that are to be set.
3. Press the LOAD button located on the front panel. This places the pattern of ones and zeros represented by the switches into the specified register, and causes the 24 console lamps to be illuminated in the same pattern.

General Specifications

The T register is used to specify changes in the system software and firmware selected by the clear/start program, as well as to request a memory dump. Each of the changes requested is specified by setting one or more bits in the T register during step (7) of the clear/start operating procedure, as follows:

Bit	Function Requested
0	Memory dump. Refer to the B 1000 Systems System Operation Guide, Volume 2, for additional information.
1	Not used.
2	Not used.
3	Not used.
4	Select the experimental MCP (MX entry), rather than the default MCP (M or MT entry).
5	Select the experimental System Initializer (NX entry), rather than the standard System Initializer (N entry).
6	Select the experimental SDL2 Interpreter (IX entry), rather than the default SDL2 Interpreter (I1 entry).
7	Select the experimental GISMO (GX entry), rather than the default GISMO (G or GT entry).
8	Select the experimental MICRO-MCP (MMX entry), rather than the standard MICRO-MCP (MM entry).
9	Select the experimental Controller (CX entry), rather than the standard NDL Controller (C entry).
10	Select the experimental MCS (MCX entry), rather than the standard MCS program (MCS entry).
11	Select the experimental SYSTEM/ODT (ODX entry), rather than the standard SYSTEM/ODT program (ODT entry).
12	Enable debug halts in System Initializer.
13 - 15	Not used.
16	Initiates the MCS program that is in the MCS Name Table entry if the AMCS system option is reset. Does not initiate the MCS program if the AMCS system option is set.
17 - 23	Not used.

HOW TO SET THE OPERATING SYSTEM OPTIONS

The following paragraphs describe how to set the operating system options.

Date and Time

After the clear/start operation on the operating system is complete, the operator must enter the date and time. This is accomplished by entering the DR (date) and TR (time) system commands. This is required only if the requisite TIME/DATE system options are set.

Example:

Operator Enters:
DR 7/23/84;

ODT Output:
DATE July/23/84 MONDAY (JULIAN DATE = 84205)

Operator Enters:
TR 1506;

ODT Output:
TIME = 15:06:00.2

MCP Options

The MCP options enable the operating system to perform certain functions. The system operator can use the SO command to set an MCP option and the RO command to reset an MCP option, except in the case of the LOG and ODTL options, which are independently set with the SL command.

At coldstart time, all of the MCP options, with the exception of BOJ, DATE, DUMP, EOJ, TIME, and WFL are reset and must be set, if desired, as part of the operation of the operating system.

The DATE and TIME options are set automatically at coldstart time. The date and time must be entered after the clear/start operation before the operating system allows programs to be scheduled. If the LOG option is not set, these option can be reset, thereby making it unnecessary to enter the date and time after each clear/start operation. After a clear/start operation, the MCP options remain in the same state, set or reset, as they were before the clear/start operation was performed.

The following is a list of the available MCP options. Each is described in the following paragraphs.

AMCS	BOJ	BREL	CHRG	CLOS	COPY
DATE	DEBUG	DISP	DUMP	EOJ	FLMP
LAB	LIB	LOG	MEM	MPRI	ODTL
OPEN	PBD	PBT	RMOV	RMSG	SCHM
SQRM	TERM	THR	TIME	TRMD	VLCP
VLIO	WFL	ZIP			

AMCS

The AMCS option causes the Message Control System (MCS) program specified in the MCS slot of the MCP NAME TABLE to be automatically executed after performing the clear/start operation.

BOJ

The BOJ option specifies that a beginning-of-job (BOJ) message be displayed each time the MCP initiates an executable object program.

BREL

The BREL option causes the MCP to display the name of a printer or punch backup file when the backup file is closed and available for printing or punching. If the BREL option is reset, no message is displayed when a backup file is closed.

CHRG

The CHRG option requires that all program executions be accompanied with a charge number, which is entered in the log. Once set, the CHRG option cannot be reset.

CLOS

The CLOS option specifies that the MCP display a file closed message each time an object program closes a file. The message has the following form.

<file-name> CLOSED <close-options>

COPY

The COPY option causes information about files being transferred by the SYSTEM/COPY program to be displayed on the ODT. The message has one of the following forms, depending on whether a COPY, COMPARE, or COPY AND COMPARE command was entered.

<file-name> COPIED
<file-name> COMPARED
<file-name> COPIED AND COMPARED

DATE

The DATE option is set at coldstart time and specifies that the ** DR PLEASE message be displayed at the end of each clear/start operation. When the ** DR PLEASE message is displayed, the system operator must enter the date with the DR command before program execution can begin. The DATE option cannot be reset if the LOG option is set.

DEBUG

The DEBUG option enables certain additional system commands and MCP functions intended for system software development and debugging. These debugging functions are not required for normal system operation.

DISP

The DISP option causes the MCP to include code segment/displacement or page/segment/displacement information in the abnormal termination message for a program. When this option is reset, the termination information is not displayed.

DUMP

The DUMP option must be set in order to dump system memory. If the DUMP option is reset, the SYSTEM/DUMPFIL file is removed from the SYSTEM disk and the space made available to the system. Any attempt to dump system memory (not the DM or DP for a program) is ignored by the MCP if the DUMP option is reset. A firmware halt with L = @00F017@ occurs if a clear/start memory dump is attempted with the DUMP option reset.

EOJ

The EOJ option specifies that the end-of-job (EOJ) message be displayed each time an object program reaches normal end of job.

FLMP

The FLMP option causes the SYSTEM/INIT program to retain code in GISMO to enable the fixed lamp display. A clear/start operation must be done to invoke the FLMP option. Refer to appendix B for a complete description of the fixed lamp display.

LAB

The LAB option causes the MCP to display a label name when a Tape Unit or Disk Pack becomes ready. The character set for a Train Printer is also displayed.

LIB

The LIB option causes the MCP to display library maintenance actions performed on disk files. The message displayed on the ODT can be one of the following when a file title has been changed, a file has been copied, a file has been removed, or a file has been updated or replaced.

```
<file-name> CHANGED TO <file-name>  
<file-name> REMOVED  
<file-name> REPLACED
```

LOG

The LOG option causes the MCP to keep a log of all program executions on disk. Refer to descriptions of the LG, SL, and TL commands in this section for actions pertaining to the LOG option. The LOG option cannot be referenced by the RO or SO MCP options. When the LOG option is set, the names of any programs that were aborted by a system halt are displayed on the ODT when a clear/start operation is performed.

When the LOG option is set, the MCP automatically provides information for the Time Analysis and Billing System (TABS) and for several other Burroughs-supplied log-analysis utilities. The following restrictions apply to the LOG option.

- To set or reset the LOG option, there must not be any tasks in the mix or in either schedule.
- After the LOG option is set, a clear/start operation is required for the setting to take effect.
- When the LOG option is set, the DATE and TIME options are unconditionally set by the MCP.

MEM

The MEM option, when set, causes the MCP to display messages regarding insufficient memory conditions.

MPRI

The MPRI option, when set, causes the SYSTEM/INIT program to retain the code in GISMO necessary to implement Level Three (Priority Memory Management) of the memory management system. Refer to appendix A for a complete description of the Level Three memory management system. A clear/start operation is required after setting or resetting the MPRI option for the MPRI option to take effect.

ODTL

The ODTL option requests that the SYSTEM/ODT program maintain a log on disk of all system commands and output ODT messages. Refer to the LG, SL, and TL commands in this section for information concerning the ODTLOG file. The ODTL option cannot be referenced by the RO and SO MCP system control instructions.

OPEN

The OPEN option causes the MCP to display a message each time an object program opens a file. The open message has the following form.

<file-name> OPENED <open-options>

PBF

PBD

The PBD option causes output files assigned to a printer or card punch to be diverted to a disk backup file if the required output device is not available when the object program opens the file.

PBT

The PBT option causes output files assigned to a printer or card punch to be diverted to a tape backup file if the required output device is not available when the object program opens the file.

NOTE

If both the PBD and PBT options are set and both the BACKUP.DISK and BACKUP.TAPE file attributes are set in the program, then backup files go to tape if a tape unit is available. If a tape unit is not available, the backup file goes to disk.

RMOV

The RMOV option, when set, automatically removes the old file in DUPLICATE FILE ON DISK situations as though an RM command had been entered by the system operator.

RMSG

The RMSG option, when set, causes the MCP output message directed to remote terminals also to be displayed on the ODT. Remote messages are not displayed at the ODT if the RMSG option is reset, except for error messages that require attention of the system operator.

SCHM

The SCHM option causes the MCP to display a message when a program is placed in the schedule. The message has the following form.

<mix-number> <program-name> NEEDS <integer> KB,
SCHED PR = <schedule-priority> IN FOR
<hours>:<minutes>:<seconds>.<tenths of a second>,
<number-of-levels> DEEP IN ACTIVE SCHEDULE

SQRM

The SMCP Queue and Remote Message (SQRM) option, when set, causes the code in the MICRO-MCP that handles queue and remote messages to be bypassed. This results in all such communications being processed exclusively by the SMCP. When the SQRM option is reset, queue and remote messages are handled by the MICRO-MCP. However, the code from both the SMCP and MICRO-MCP can be resident in memory at certain times to handle exception conditions. In cases where system usage is extremely heavy and memory space is limited, an increase in throughput can be realized by setting the SQRM option.

TERM

The TERM option, when set, causes the MCP to automatically discontinue processing (DS) a program when an error condition is encountered. If an error condition occurs and it is necessary to automatically obtain a memory dump of the program, the TERM option must be reset, or the TRMD option must be set.

THR

The THR option, when set, causes the SYSTEM/INIT program to retain code in GISMO necessary to implement Level Two (Thrashing Detection) of the memory management system. Refer to appendix A for a complete description of the Level Two memory management system. A clear/start operation is required after setting or resetting this option in order for the THR option to take effect. If the MPRI option is set, the GISMO code necessary to perform Thrashing Detection is retained regardless of the setting of the THR option.

TIME

The TIME option, when set, causes the ** TR PLEASE message to be displayed on the ODT at the end of a clear/start operation. The TIME option is set at coldstart time. When the ** TR PLEASE message is displayed, the system operator must enter the time with the TR command before program execution can begin. Refer to the TR command in this section for complete information. The TIME option cannot be reset if the LOG option is set.

TRMD

The TRMD option, when set, causes the MCP to automatically dump memory and discontinue processing (DP) of a program when an error condition is encountered. If both the TERM and TRMD options are set, the TRMD option takes precedence.

VLCP

The VLCP option, when set, causes the SYSTEM/INIT program to retain code in GISMO to enable displaying of processor usage and overlay activity. Refer to appendix B for complete information concerning the performance monitoring system. A clear/start operation is required to invoke the VLCP option.

VLIO

The VLIO option, when set, causes the SYSTEM/INIT program to retain code in GISMO to enable the display of activity on the I/O subsystem. Refer to appendix B for complete information concerning the performance monitoring system. Setting the VLIO system option, causes the VLCP code to be included as well. A clear/start operation is required to invoke the VLIO option.

WFL

The WFL option, when set, causes WFL syntax to be used for processing ambiguous system commands; otherwise Control Card (CC) syntax is used. The default for the WFL option is SET. Several ODT-Commands, such as CHANGE, COMPILE, MODIFY, and REMOVE have two syntax diagrams. If these commands are prefixed with WFL, then WFL syntax is used; if these commands are prefixed with CC then Control Card syntax is used. When a prefix is not used, then the WFL option determines the syntax used.

ZIP

The ZIP option, when set, causes all programmatic ZIP statements to be displayed on the ODT.

INSTALLING NEW SYSTEM SOFTWARE UPDATES

The following is a suggested procedure for loading new system files. If some of the system software is not to be replaced, then those files may be omitted in the following steps.

Determine Current Operating Environment

Determine the names of the GISMO, MCP II, Micro MCP, SDL2 interpreter, SYSTEM/COPY, SYSTEM/INIT, SYSTEM/ODT, SYSTEM/PANDA, and the NDL handler files currently running on the system, by using the WW keyboard command as follows:

```
WW G/=;  
WW M/=;  
WW MM/=;  
WW I/=;  
WW CPY;  
WW N/=;  
WW O/=;  
WW PAN/=;  
WW C/=;
```

The Name Table entries of concern are the G, M, MM, I, CPY, N, ODT, PAN, and C entries. In the following steps, it is assumed that the files in these entries have the names GISMO3, MCP II, MCP II/MICRO-MCP, SDL2/INTERP3M, SYSTEM/COPY SYSTEM/INIT, SYSTEM/ODT, SYSTEM/PANDA, and SYSTEM/CONTROLLER respectively. If the names are not the same, the different name may be substituted in the examples (such as substituting MMCP/DEBUG for MCP II/MICRO-MCP).

Copy the Software to be Replaced

Use the COPY command to make a copy of the current files to be replaced GISMO3, MCPPII, MCPPII/MICRO-MCP, SDL2/INTERP3M, SYSTEM/INIT, SYSTEM/ODT, SYSTEM/PANDA, and SDL2INTRIN/AGGREGATE, to files named OLD/GISMO3, OLD/MCPPII, OLD/MICRO-MCP, OLD/SDL2INT, OLD/SYSINIT, OLD/SYSODT, OLD/SYSPAN, and SDL2OLD/AGGREGATE respectively, as follows:

```
COPY GISMO3 AS OLD/GISMO3,  
    MCPPII AS OLD/MCPPII,  
    MCPPII/MICRO-MCP AS OLD/MICRO-MCP,  
    SDL2/INTERP3M AS OLD/SDL2INT,  
    SYSTEM/INIT AS OLD/SYSINIT,  
    SYSTEM/ODT AS OLD/SYSODT,  
    SYSTEM/PANDA AS OLD/SYSPAN  
    SDL2INTRIN/AGGREGATE AS SDL2OLD/AGGREGATE  
FROM DISK TO DISK;
```

Use Experimental Entries

Use the CM keyboard command to place the names of the current files, GISMO3, MCPPII, MCPPII/MICRO-MCP, SDL2/INTERP3M, SYSTEM/INIT, and SYSTEM/ODT into the experimental entries of the Name Table (as a precautionary measure) and the names of the copied files, OLD/GISMO, OLD/MCPPII, OLD/MICRO-MCP, OLD/SDL2INT, OLD/SYSINIT, and OLD/SYSODT, into the standard entries of the Name Table, as follows:

```
CM GX GISMO;  
CM MX MCPPII;  
CM MMX MCPPII/MICRO-MCP;  
CM IX SDL2/INTERP3M;  
CM NX SYSTEM/INIT;  
CM ODX SYSTEM/ODT;  
CM G OLD/GISMO;  
CM M OLD/MCPPII;  
CM MM OLD/MICRO-MCP;  
CM I OLD/SDL2INT;  
CM N OLD/SYSINIT;  
CM ODT OLD/SYSODT;  
CM PAN OLD/SYSPAN
```

Loading a New SDL2 Interpreter

If loading a new SDL2 interpreter, the programs that must be running during this process must be modified to use an interpreter named MCP/INTERP. Therefore, modify the copied SYSTEM/ODT file, the SYSTEM/PANDA file, the SYSTEM/COPY file, and the NDL handler to use an interpreter named MCP/INTERP. When the MCP encounters MCP/INTERP, the interpreter that the program uses is made the same as the interpreter that the MCP is using. For example:

```
MODIFY OLD/SYSODT INTERPRETER MCP/INTERP;  
MODIFY SYSTEM/COPY INTERPRETER MCP/INTERP;  
MODIFY SYSTEM/CONTROLLER INTERPRETER MCP/INTERP;  
MODIFY OLD/SYSPAN INTERPRETER MCP/INTERP;
```


Loading a New SDL2 Intrinsic File

If loading a new SDL2 intrinsic file, the programs that must be running during this process must be modified to use the copied intrinsic file. For example:

```
MODIFY OLD/SYSODT INTRINSIC.NAME SDL2OLD;  
MODIFY SYSTEM/COPY INTRINSIC.NAME SDL2OLD;  
MODIFY SYSTEM/CONTROLLER INTRINSIC.NAME SDL2OLD;  
MODIFY OLD/SYSPAN INTRINSIC.NAME SDL2OLD;
```

clear/start With OLD/Software

Perform a clear/start operation. The currently running GISMO, MCP II, Micro MCP, SDL2 interpreter, SYSTEM/INIT, SYSTEM/ODT, and SYSTEM/PANDA files are now OLD/GISMO3, OLD/MCP II, OLD/MICRO-MCP, OLD/SDL2INT, OLD/SYSINIT, OLD/SYSODT, and OLD/SYSPAN, respectively.

Remove Old System Files

Remove the now unused files, GISMO3, MCP II, MCP II/MICRO-MCP, SDL2/INTERP3M, SYSTEM/INIT, SYSTEM/ODT, and SYSTEM/PANDA from the system disk, as follows:

```
CM GX PURGE;  
CM IX PURGE;  
CM MX PURGE;  
CM MMX PURGE;  
CM NX PURGE;  
CM ODX PURGE;  
REMOVE GISMO3;  
REMOVE SDL2/INTERP3M;  
REMOVE MCP II;  
REMOVE MCP II/MICRO-MCP;  
REMOVE SYSTEM/INIT;  
REMOVE SYSTEM/ODT;  
REMOVE SYSTEM/PANDA
```

NOTE

After removal of the SDL2/INTERP3M file, system programs (such as DMPALL) and compilers should not be run, except for OLD/SYSODT, OLD/SYSPAN, SYSTEM/COPY, and the modified NDL handler until the new SDL2 interpreter is copied.

Copy New System Files

Use the COPY AND COMPARE command to copy the new files, GISMO3, MCP II, MCP II/MICRO-MCP, SDL2/INTERP3M, SYSTEM/INIT, SYSTEM/ODT, and SYSTEM/PANDA to the system disk from either the disk or tape media provided. Also, if loading a new SDL2 intrinsic file, then copy the SDL2INTRIN/REMOVER program to the system disk. For example:

```
COPY AND COMPARE GISMO3, MCP II, MCP II/MICRO-MCP,  
SDL2/INTERP3M, SYSTEM/INIT, SYSTEM/ODT, SYSTEM/PANDA  
SDL2INTRIN/REMOVER FROM <media> TO DISK;
```

Perform the CM(s)

Use the CM keyboard command to place the names of the current GISMO, MCP II, Micro MCP, SDL2 interpreter, SYSTEM/INIT, and SYSTEM/ODT files into the experimental entries of the Name Table (as a precautionary measure) and the names of the new files, GISMO3, MCP II, MCP II/MICRO-MCP, SDL2/INTERP3M, SYSTEM/INIT, SYSTEM/ODT, and SYSTEM/PANDA into the standard entries of the Name Table, as follows:

```
CM GX OLD/GISMO3;
CM IX OLD/SDL2INT;
CM MX OLD/MCP II;
CM MMX OLD/MICRO-MCP;
CM NX OLD/SYSINIT;
CM ODX OLD/SYSODT;
CM G GISMO3;
CM I SDL2/INTERP3M;
CM M MCP II;
CM MM MCP II/MICRO-MCP;
CM N SYSTEM/INIT;
CM ODT SYSTEM/ODT;
CM PAN SYSTEM/PANDA
```

It is important that the new SDL2 interpreter that is CMed into the I Name Table Entry has the name SDL2/INTERP3M.

SYSTEM/COPY and SDL2 Intrinsic File

If loading a new SYSTEM/COPY or a new SDL2 intrinsic file, then use the COPY AND COMPARE command to copy the new SYSTEM/COPY file as NEW/SYSCOPY, and the new SDL2 intrinsic file as SDL2NEW/AGGREGATE. For example:

```
COPY AND COMPARE SYSTEM/COPY AS NEW/SYSCOPY,
SDL2INTRIN/AGGREGATE AS SDL2NEW/AGGREGATE
FROM <media> TO DISK;
```

Loading SYSTEM/COPY

If loading a new SYSTEM/COPY, then perform the following sequence of operations.

```
CM CPY NEW/SYSCOPY
CHANGE SYSTEM/COPY TO OLD/SYSCOPY;
COPY NEW/SYSCOPY AS SYSTEM/COPY;
CM CPY SYSTEM/COPY;
REMOVE NEW/SYSCOPY
```

Loading SDL2 Intrinsic File

If loading a new SDL2 intrinsic file, then execute the SDL2INTRIN/REMOVER program to mark the current SDL2INTRIN/AGGREGATE file as an unrestricted file, as follows:

```
EXECUTE SDL2INTRIN/REMOVER;
```

Remove the current SDL2INTRIN/AGGREGATE file, and change the name of the new SDL2INTRIN/AGGREGATE file to the proper name, as follows:

```
REMOVE SDL2INTRIN/AGGREGATE;  
CHANGE SDL2NEW/AGGREGATE TO SDL2INTRIN/AGGREGATE;
```

Execute the SDL2INTRIN/REMOVER program to mark the new SDL2INTRIN/AGGREGATE file as a restricted system file, as follows:

```
EXECUTE SDL2INTRIN/REMOVER;
```

NDL Handler

If the NDL handler was modified to a new SDL2 intrinsic name, then modify the intrinsic name back to the proper name. For example:

```
MODIFY SYSTEM/CONTROLLER INTRINSIC.NAME SDL2INTRIN;
```

Clear/Start the New System Software

Perform a clear/start operation to bring up the new system software.

Remove Old Software

The now unused files, OLD/GISMO3, OLD/MCPH, OLD/MICRO-MCP, OLD/SDL2INT, OLD/SYSCOPY, OLD/SYSINIT, OLD/SYSODT, OLD/SYSPAN, SDL2INTRIN/AGGREGATE can now be removed from the SYSTEM disk. For example:

```
CM GX PURGE;  
CM IX PURGE;  
CM MX PURGE;  
CM MMX PURGE;  
CM NX PURGE;  
CM ODX PURGE;  
REMOVE OLD/GISMO3;  
REMOVE OLD/SDL2INT;  
REMOVE OLD/MCPH;  
REMOVE OLD/MICRO-MCP;  
REMOVE OLD/SYSCOPY;  
REMOVE OLD/SYSINIT;  
REMOVE OLD/SYSODT;  
REMOVE OLD/SYSPAN;  
REMOVE SDL2OLD/AGGREGATE;
```

Experimental Name Table Entries

If the system normally has files in the experimental Name Table entries (such as MICRO-MCP/DEBUG in the MMX entry), the files can be copied to the system disk and entered into the appropriate entry in the Name Table with the CM system command.

SECTION 4

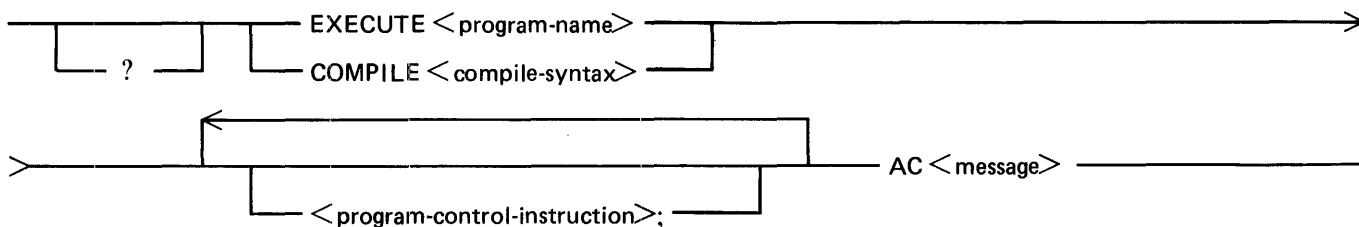
PROGRAM CONTROL INSTRUCTIONS

This section contains a description of each program control instruction. These instructions are used to control the compilation, execution, modification, and query of programs.

AC

The AC program control instruction enables an early response to an ACCEPT message to be entered at the time a program is compiled or executed. It functions exactly like the AX program control instruction; however, it cannot be delimited with a semicolon (;) character. If a semicolon (;) character follows the AC keyword, it is treated as if it is a part of the early response.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

message

This field can contain any group of alphanumeric characters and is a message written to the program. <message> starts in the first position after the AC keyword. If <message> is shorter than the receiving field in the program, <message> is padded on the right with blank characters. If <message> is longer than the receiving field in the program, <message> is truncated on the right.

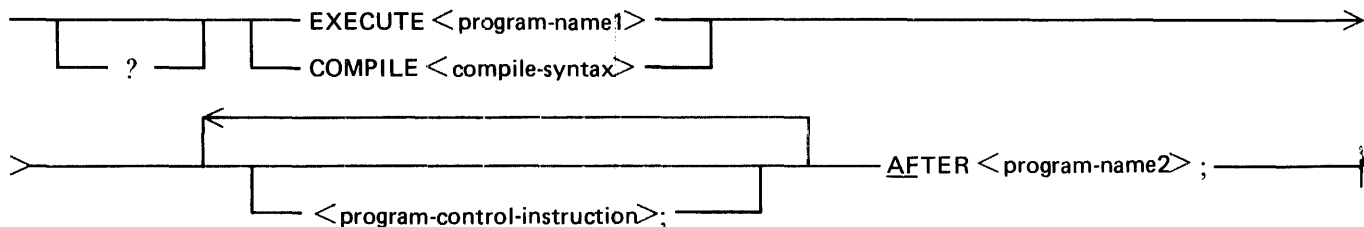
Example:

```
EXECUTE DMPALL;AC LIST TEST/PROGRAM ALPHA
```

AFTER

The AFTER program control instruction conditionally schedules a program after the termination of another program.

Syntax:



Semantics:

program-name1

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

program-name2

This field can contain any valid B 1000 program name and specifies the name of an executing or scheduled to be executed program. The program execution specified by <program-name2> or the compile specified by <compile-syntax> must terminate before <program-name1> is performed.

Example:

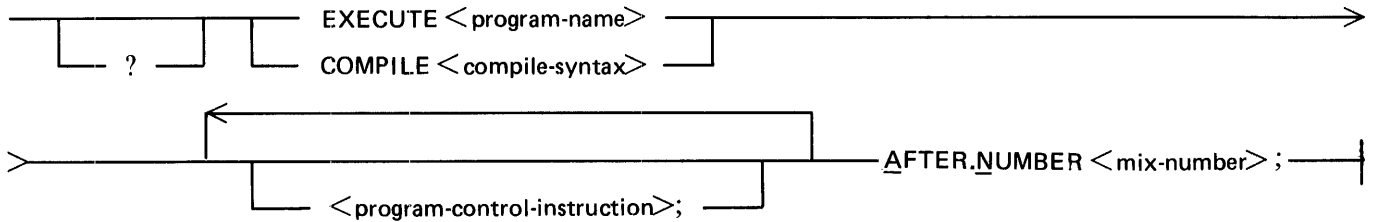
```
EXECUTE ALPHA AFTER BETA;    % When the program BETA goes to end
                             % of job, the program ALPHA is placed
                             % in the active schedule for execution
                             % as soon as memory resources are
                             % available or the system mix
                             % limit permits.
```

AFTER.NUMBER

AFTER.NUMBER

The AFTER.NUMBER program control instruction schedules a program after the termination of another program by mix number.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program. The program specified by <mix-number> must terminate before the execution specified by <program-name> or the compile specified by <compile-syntax> is performed.

Examples:

```

EXECUTE ALPHA AFTER.NUMBER 7;    % When the program (mix-number 7)
                                  % goes to end of job, the program
                                  % ALPHA is placed in the active
                                  % schedule for execution as soon
                                  % as memory resources are available
                                  % or the system mix limit permits.
    
```

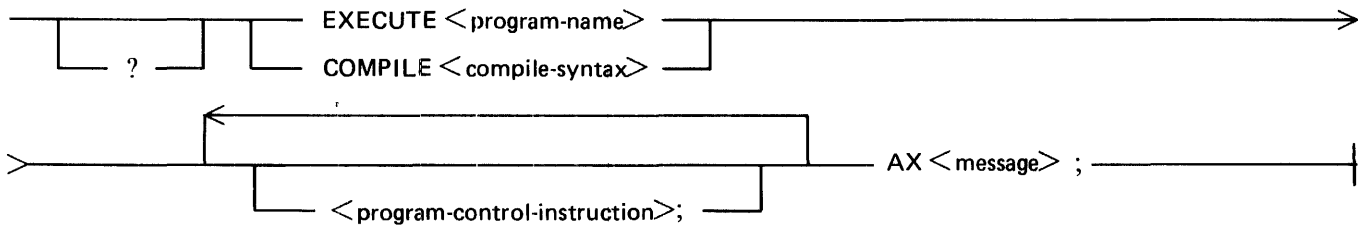
```

EX ALPHA AN 8;
    
```

AX

The AX program control instruction enables an early response to an ACCEPT message to be entered at the time a program is compiled or executed. It functions exactly like the AC program control instruction; however, the semicolon (;) character can be used as a delimiter. If a semicolon (;) character follows the AX keyword, it terminates the early response and additional program control instructions can follow. The semicolon (;) character is not passed to the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

message

This field can contain any group of alphanumeric characters and is a message written to the program. <message> starts in the first position after the AX keyword. If <message> is shorter than the receiving field in the program, <message> is padded on the right with blank characters. If <message> is longer than the receiving field in the program, <message> is truncated on the right.

Example:

```
EXECUTE DMPALL;AX LIST TEST/PROGRAM ALPHA;AX LIST TEST/PAYROLL;
```

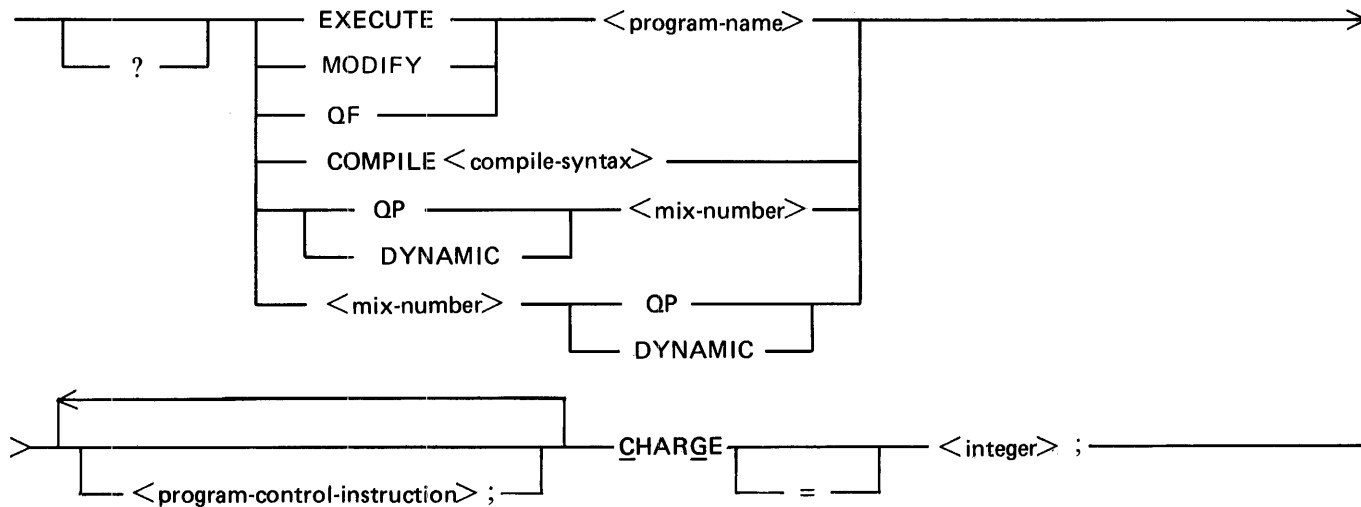

CHARGE

CHARGE

The CHARGE program control instruction inserts a charge number into the log record for a program.

If the MCP CHARGE option is set, the CHARGE program control instruction must be specified before a program is scheduled.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any 1- to 7-digit number and specifies the charge number to be inserted in the MCP log file. If less than seven digits are specified, leading zeroes are assumed.

Examples:

```
EXECUTE DMPALL CHARGE 31000;
```

```
MODIFY PROG1 CG 154;
```

CLASS

COMPILE

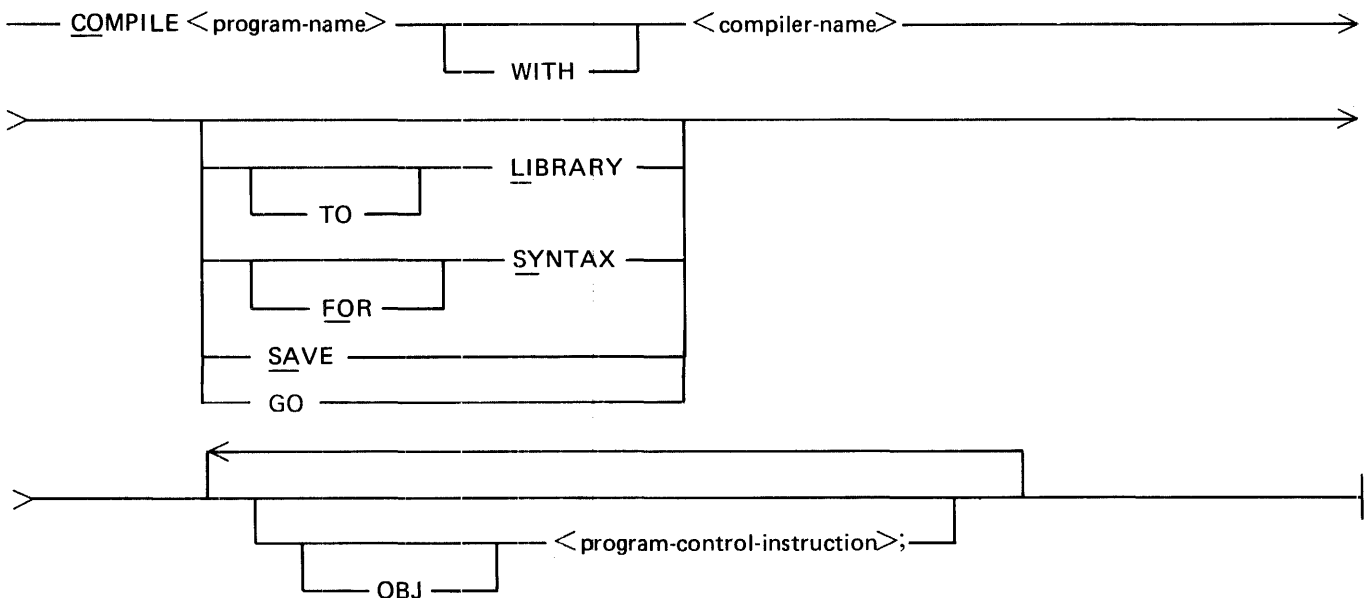
The COMPILE program control instruction designates the compiler to be used and the type of compilation to be performed.

The syntax of the unabbreviated COMPILE program control instruction is determined by the setting of the WFL system option. For more information see the description of the WFL option in section 3. The syntax for the WFL version of the COMPILE program control instruction is in the B 1000 Systems WFL Language Manual.

The COMPILE program control instruction has four options.

1. The compile and go operation compiles the object program, executes the object program, and does not enter <program-name> in the disk directory. If the LIBRARY, SAVE, or SYNTAX keywords are not specified in the COMPILE instruction, the compile and go operation is invoked.
2. The compile to library operation compiles the object program and enters <program-name> into the disk directory. The LIBRARY keyword must be specified in the COMPILE instruction to invoke the compile to library operation.
3. The compile and save operation compiles the object program, executes the object program, and enters <program-name> into the disk directory. The SAVE keyword must be specified in the COMPILE command to invoke the compile and save operation.
4. The compile for syntax operation causes the compile to be for syntaxing only. This provides a compiler-generated listing for debugging, a first time compilation, or a new source listing.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 object program name that follows the B 1000 file-naming conventions.

compiler-name

COMPILE

This field can contain any valid B 1000 compiler name. BASIC, IBASIC, COBOL, COBOL74, FORTRAN, FORTRAN77, RPG, UPL, and UPL2 are examples of valid B 1000 compiler names.

LIBRARY

The LIBRARY keyword causes <program-name> to be entered in the disk directory if the compilation completes without a syntax error. The object program is not scheduled for execution.

SYNTAX

The SYNTAX keyword causes the compiler to check for syntax errors but not to generate an object program. This provides a diagnostic listing as output to be used as a debugging tool, a first time compilation, or a new source listing. <program-name> is not entered in the disk directory.

SAVE

The SAVE keyword causes <program-name> to be entered in the disk directory and executes the object program if the compilation completes without a syntax error.

GO

The GO keyword causes the object program to be executed if the compilation completes without a syntax error. The program, <program-name>, is not entered in the disk directory.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow a COMPILE program control instruction.

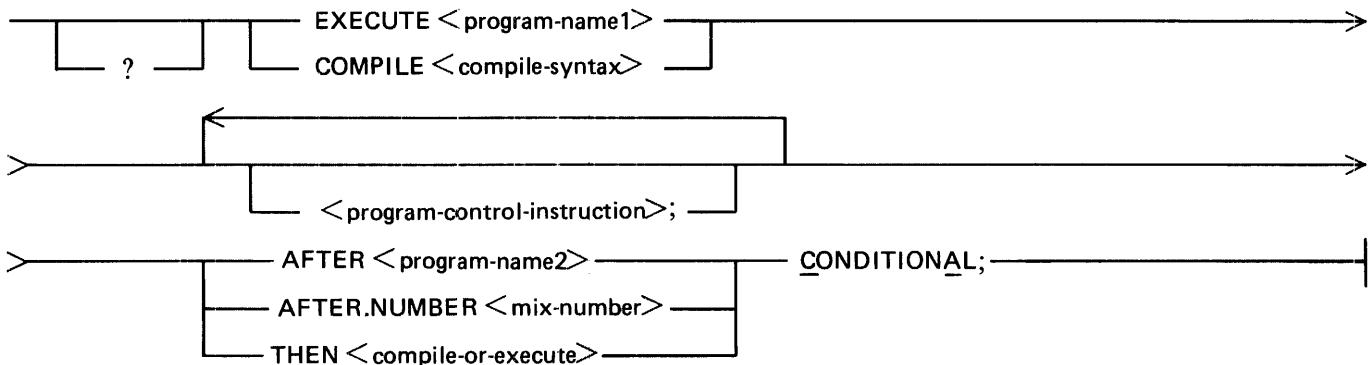
OBJ

The OBJ keyword represents the OBJ program control instruction. It specifies that the <program-control-instruction> that follows applies to the object program specified by <program-name>. If the OBJ keyword is not present, the <program-control-instruction> that follows applies to the compiler specified by <compiler-name> instead of the object program. All <program-control-instruction>s preceded by the OBJ keyword are queued by the MCP and, following successful compilation, are used to perform an automatic MODIFY program control instruction on the object program. Only those <program-control-instruction>s that are allowed to follow the MODIFY program control instruction can be used following the OBJ keyword.

CONDITIONAL

The **CONDITIONAL** program control instruction inhibits a program from being executed unless its predecessor successfully reaches end of job without an error. This instruction is used in conjunction with the **AFTER**, **AFTER.NUMBER**, and **THEN** program control instructions. The **CONDITIONAL** program control instruction is a default instruction. The **UNCONDITIONAL.EXECUTION** program control instruction can be used to change the default.

Syntax:



Semantics:

program-name1

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the **COMPILE** keyword, refer to the **COMPILE** program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the **EXECUTE** or **COMPILE** instruction.

program-name2

This field can contain any valid B 1000 program name and specifies the name of an executing or scheduled to be executed program. The program specified by <program-name2> must terminate without an error before the execution specified by <program-name1> or the compile specified by <compile-syntax> is performed.

mix-number

This field can contain any mix number of an executing or scheduled to be executed program. The program specified by <mix-number> must terminate without an error before the execution specified by <program-name1> or the compile specified by <compile-syntax> is performed.

compile-or-execute

This field can contain any valid **COMPILE** or **EXECUTE** program control instruction.

Examples:

```
EXECUTE A/B AFTER C/D; CONDITIONAL;
```

```
EXECUTE A/B; AF C/D; CA;
```

DATA

DATA

The DATA program control instruction informs the MCP of the name of a punched card data file.

When a DATA program control instruction follows immediately after a COMPILE or EXECUTE program control instruction, the MCP saves the card reader device for the program specified in the COMPILE or EXECUTE instruction. This prevents any other program in the mix that requests a card file with the file identifier specified in the DATA program control instruction from being assigned the card file belonging to the program just executed. The MCP does not reserve the card reader device in this manner if an END program control instruction is detected before the DATA program control instruction.

Syntax:

 DATA <file-identifier>;

Semantics:

file-identifier

This field can contain any two-name file identifier and specifies the name of the card file that immediately follows the DATA program control instruction.

Example 1:

```
?EXECUTE A/B CHARGE 123456;  
?DATA CARDIN  
.  
.(data cards)  
.  
?END
```

Example 2:

```
?COMPILE TEST/PROGRAM COBOL74 LIBRARY DATA CARDS  
.  
.(data cards)  
.  
?END
```

Example 3:

```
?EXECUTE TEST/PROGRAM MEMORY 12000;  
?CHARGE 12666;  
?FILE CARDS NAME = TEMP DISK;  
?END  
?DATA CARDFILE  
.  
.(data cards)  
.  
?END
```

DYNAMIC

The DYNAMIC program control instruction modifies the working copy of an object program that is executing or is scheduled for execution.

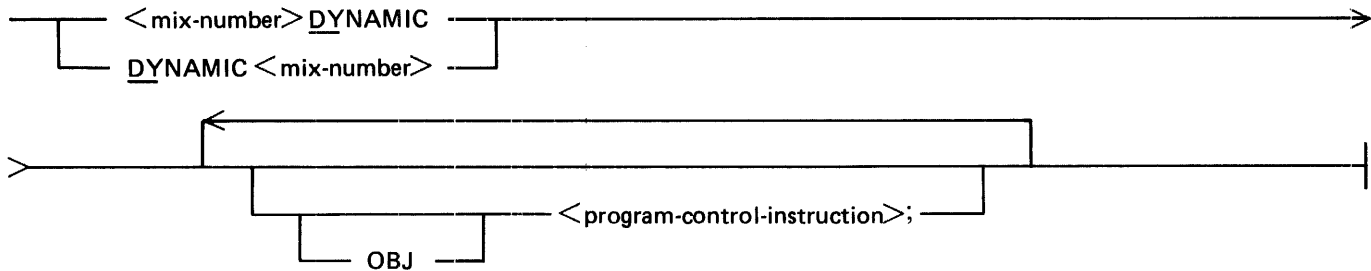
The following program control instructions are allowed to be used with the DYNAMIC instruction while an object program is scheduled for execution and has not reached beginning of job (BOJ).

CHARGE	OVERRIDE
CONDITIONAL	PRIORITY
DYNAMIC.SPACES	PROCESSOR.PRIORITY
FILE	SECONDS.BEFORE.DECAY
FREEZE	SWITCH
INTERPRETER	TIME
INTRIN.DIRECTORY	UNCONDITIONAL
INTRINSIC.NAME	UNFREEZE
MEMORY.PRIORITY	UNOVERRIDE
NODIF	VIRTUAL.DISK
OBJIF	

After an object program has reached beginning of job (BOJ), the only program control instructions that have any effect are those that are specified using the FILE program control instruction. In addition, the file being referenced in the FILE program control instruction must be completely closed with the RELEASE, LOCK, CRUNCH, or PURGE close options. The only exceptions to this are the following FILE attributes, which can be dynamically modified while the file is open.

AUTOPRINT
 END.OF.PAGE
 INVALID.CHARACTERS
 LOCK

Syntax:



Semantics:

mix-number

This field can contain any mix number of an executing or scheduled program.

program-control-instruction

This field can contain any valid program control instruction described in this section that is allowed to follow a DYNAMIC program control instruction.

DYNAMIC

OBJ

The OBJ keyword represents the OBJ program control instruction. It specifies that the <program-control-instruction> that follows applies to the object program of a compilation. The OBJ keyword can only be used when <mix-number> refers to a compilation. All <program-control-instruction>s preceded by the OBJ keyword are queued by the MCP and, following successful compilation, are used to perform an automatic MODIFY program control instruction on the object program. Only those <program-control-instruction>s that are allowed to follow the MODIFY program control instruction can be used following the OBJ keyword.

Examples:

DYNAMIC 9171 PRIORITY = 5;SB = 20;FILE LINE LABEL.TYPE = 1;

DY 25 FILE PRINT INVALID.CHARACTERS = 3;

2000 DYNAMIC FILE TAPE UNIT.NAME = MTA;

DYNAMIC.SPACES

The DYNAMIC.SPACES program control instruction assigns additional dynamic memory, beyond that specified in the MEMORY program control instruction, for memory links that are found in the overlayable data space of a program.

The MCP assigns a value of ten at execution time if the DYNAMIC.SPACES value is zero for a task using dynamic memory. The DYNAMIC.SPACES program control instruction is normally specified only if a precisely calculated value is specified for dynamic memory; if it is not specified or provided by the MCP, the space for required memory links uses part of the precise value provided by the user, causing fewer pages of overlayable data than intended to reside in dynamic memory.

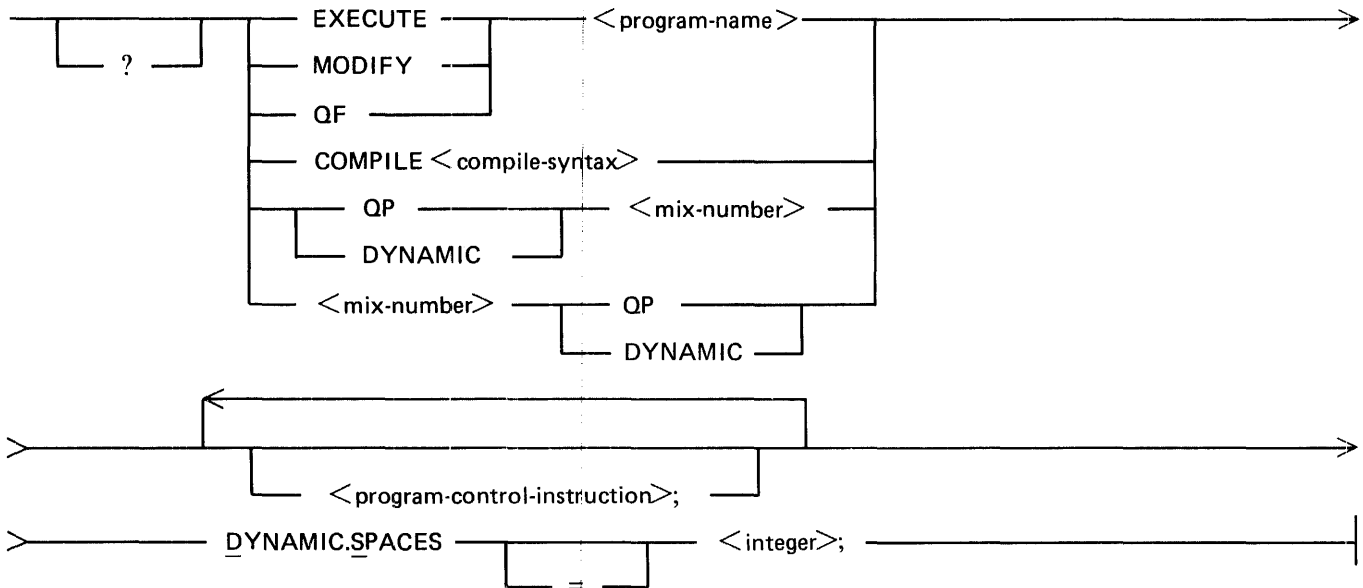
For example:

```
EXECUTE EXPORT/PAYROLL;MEMORY 21300;DYNAMIC.SPACES = 8;
```

In this example, the MCP assigns the program EXPORT/PAYROLL 21300 bits of dynamic memory plus the following:

where <memory-link-size> is currently 187 bits. The space for two extra memory links provide for the header and trailer links which delimit the overlayable data area of a program. Therefore, in the example, the MCP actually assigns $21300 + ((8 + 2) * 187) = 23170$ bits of dynamic memory to the program when it is scheduled.

Syntax:



Semantics:

DYNAMIC.SPACES

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled to be executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field specifies the number of additional memory links to be used in conjunction with programs that have dynamic memory.

END

The END program control instruction informs the MCP that the card data input has reached the end of file (EOF).

When the END program control instruction is used, it must be the last card in the file. A program receives an end-of-file report from the MCP when the program attempts to read the END program control instruction.

The END program control instruction is not required at the end of card data input if the program recognizes the last card in the file and closes that file without reading another record. If the program attempts to read another record from that file and the card reader is empty, the MCP holds the card reader waiting for more data or an END program control instruction.

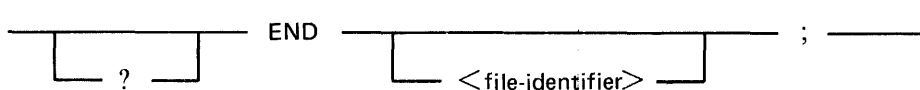
A card file need not be terminated by an END input card if it is immediately followed by another control card. The detection of this other control card by the MCP results in an end of file signal to the program, as though an END card had been read.

If a data card with an invalid punch character in column one is read, the MCP stops the card reader and notifies the operator that the card just read has an invalid punch in column one. This allows the operator to correct the card and permits the program to continue reading cards.

The END program control instruction can also separate a following DATA program control instruction from a preceding EXECUTE or COMPILE program control instruction when it is desired to disassociate the data file from the program specified in the EXECUTE or COMPILE instruction. This is the only case in which the END program control instruction can appear on the same card as other control instructions, and without a question mark (?) or invalid character in column one.

If the END program control instruction is used in text zipped to the MCP, no information following it is processed by the MCP.

Syntax:



Semantics:

file-identifier

This field can contain any valid two-name file identifier and specifies the file name of the card file in the previous DATA program control instruction.

Example 1:

```
?EXECUTE A/B;  
?DATA CARDS;  
  .  
  .(data cards)  
  .  
?END CARDS;
```

Example 2:

END

```
?COMPILE X/Y FORTRAN77 DA CARDS
```

```
.(data cards)
```

```
?END CARDS;
```

Example 3:

```
?EX TEST END;
```

```
?DATA CARDFIL;
```

```
.(data cards)
```

```
?END CARDFILE;
```

ENDCTL

The ENDCTL (End Control) program control instruction indicates to the MCP that the input file to the SYSTEM/LDCONTRL program has reached the end of the file.

The ENDCTL program control instruction must be the last card in an input file read by the SYSTEM/LDCONTRL program. This file is often called the control deck. The ENDCTL program control instruction causes the MCP to notify the SYSTEM/LDCONTRL program that the end of the control deck has been reached and closes all files in use and sends the SYSTEM/LDCONTRL program to end of job.

Refer to the LD system command in section 5 of this manual and the SYSTEM/LDCONTRL program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the SYSTEM/LDCONTRL program.

Syntax:

— ?ENDCTL; ————|

Example:

```
?DATA CTLDCK;  
?COMPILE TEST RPG LIBRARY;  
?DATA CARD;  
  .  
  .(data cards)  
  .  
?END CARD;  
?COMPILE TESTER FORTRAN77 LIBRARY;  
?DATA CARD;  
  .  
  .(data cards)  
  .  
?END CARD;  
?ENDCTL;
```

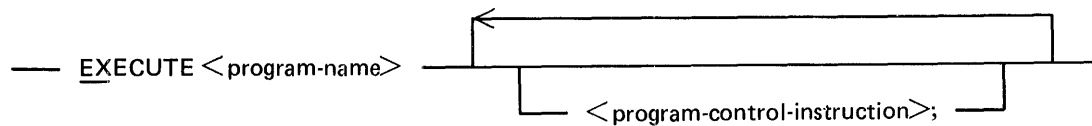
EXECUTE

EXECUTE

The EXECUTE program control instruction causes the MCP to call an object program from the library for subsequent execution. The EXECUTE program control instruction must be the first program control instruction in a set of program control instructions pertaining to the execution of an object program.

If <program-name> resides on a user disk, the disk identifier must be part of the object program in order for the MCP to locate the correct file.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 object program name that follows the B 1000 file-naming conventions and specifies the program to be executed.

program-control-instruction

This field can contain any valid program control instruction described in this section that is allowed to follow an EXECUTE program control instruction.

Examples:

```
EXECUTE TEST FILE DATA NAME MASTERFILE;
```

?EXECUTE PROG1	%0 Executes the object program labeled
?DATA CARDS	%0 PROG1 through a card reader using
.	%0 a card deck delimited by ?DATA CARDS and
(data cards)	%0 ?END as beginning and terminating
.	%0 records, respectively. The PROG1 program
?END	%0 declared a card file labeled CARDS.

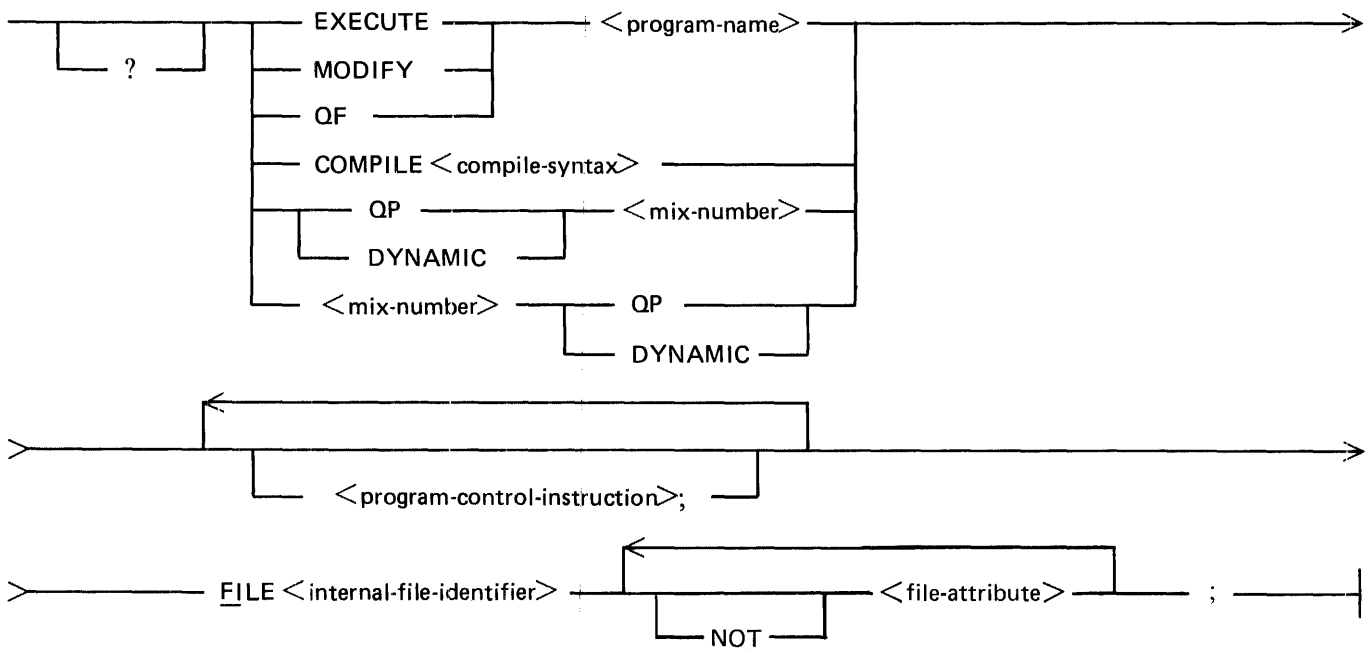
FILE

The FILE program control instruction specifies various file attributes for input and output files.

The elements within the FILE program control instruction must be separated by at least one blank character, and must be terminated with a semicolon (;) character or END OF MESSAGE. If the FILE program control instruction is entered by way of a card reader, each of the continuation cards must have a question mark (?) character in column one.

The FILE program control instruction must immediately follow the COMPILE, EXECUTE, DYNAMIC, or MODIFY program control instructions. The MCP modifies the information in a working copy of the File Parameter Block (FPB) of the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

FILE

This field can contain any mix number of an executing or scheduled to be executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B FILE MASTER NAME MASTER/NEW DISK DEFAULT;
```

```
MODIFY PROG1 FI LINE USER.BACKUP.NAME NAME #PROG1 NO HARDWARE;
```

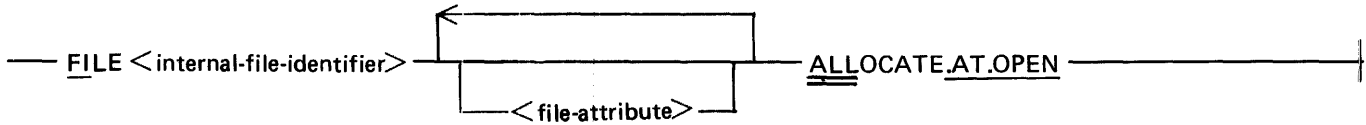
File Attributes

The following is a list of file attributes that can follow the FILE program control instruction.

ALLOCATE.AT.OPEN

The ALLOCATE.AT.OPEN file attribute causes all of the disk file areas requested by this file to be allocated at the time the file is opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER ALLOCATE.AT.OPEN;
```

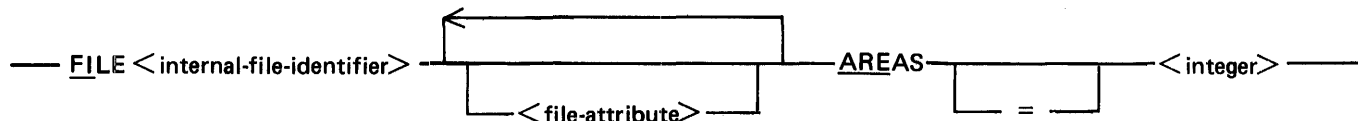
```
MODIFY PROG1; FILE PAYROLL ALL;
```


AREAS

FILE AREAS

The AREAS file attribute specifies the number of disk areas assigned to the file. The number of disk areas can range from 1 to 105 inclusive. This file attribute applies only to new files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 105 inclusive, and specifies the number of disk areas assigned to the file.

Examples:

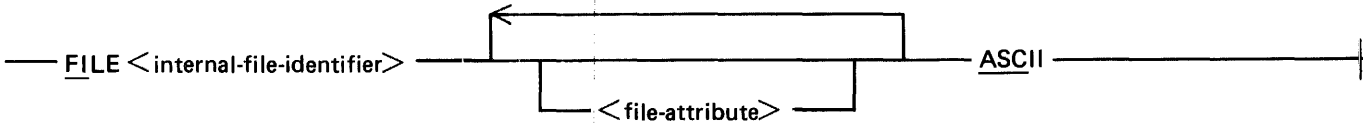
```
EXECUTE A/B; FILE MASTER AREAS = 50;
```

```
MODIFY PROG1; FILE PAYROLL ARE 105;
```

ASCII

The ASCII file attribute specifies the recording mode to be the American Standard Code for Information Interchange (ASCII) recording mode. This attribute is only applicable for magnetic tape files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE MASTER ASCII;

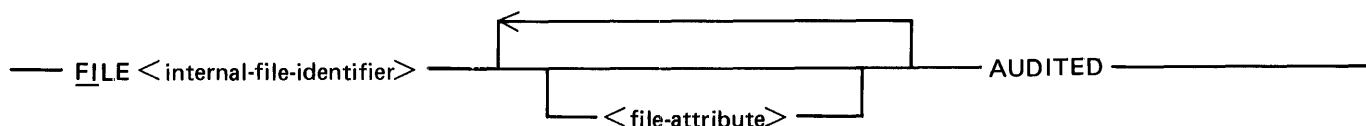
MODIFY PROG1; FILE PAYROLL ASC;

AUDITED

FILE AUDITED

The AUDITED file attribute causes a task that is performing an input/output operation on a file to be suspended until the physical operation is complete. When a task performs an output operation on an ISAM file that has the AUDITED file attribute set, the task is suspended until the physical write operations on the data file and index files are complete. If a system halt occurs during an output operation to a file that has the AUDITED file attribute set, the maximum amount of information that can be lost is one record.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

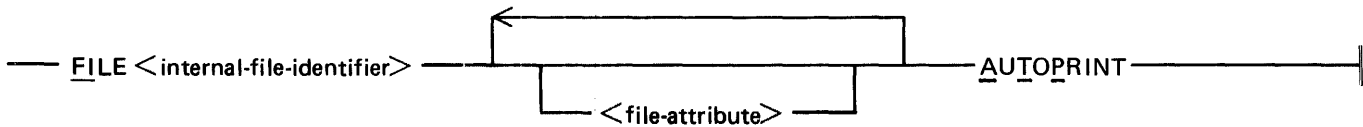
EXECUTE A/B; FILE MASTER AUDITED;

MODIFY PROG1; FILE VERIFY AUDITED;

AUTOPRINT

The AUTOPRINT file attribute causes the backup printer disk file to be entered in the MCP autoprint queue when the file is closed. This file attribute is set by default. Specifying NO AUTOPRINT prevents the backup printer disk file from being processed by the MCP autoprint mechanism.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE LINE AUTOPRINT;

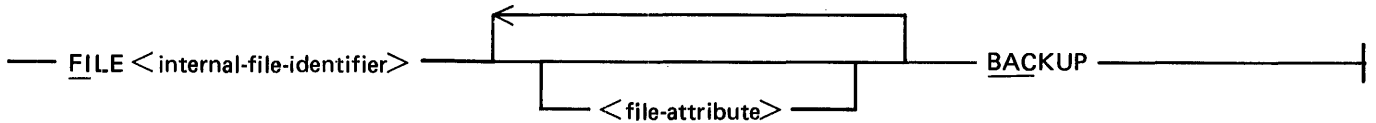
MODIFY PROG1; FILE LINE ATP;

BACKUP

FILE BACKUP

The BACKUP file attribute causes the output of the file to go to backup. This file attribute sets the BACKUP.DISK and BACKUP.TAPE file attributes for this file by default.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

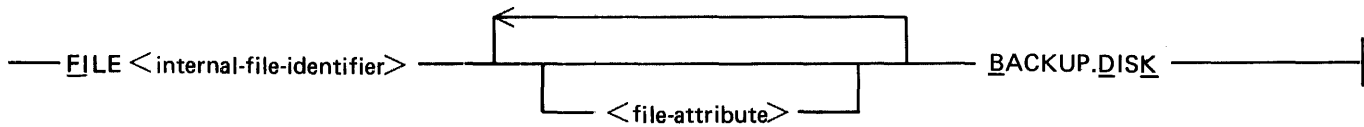
EXECUTE A/B; FILE LINE BACKUP

MODIFY PROG1; FILE LINE BAC

BACKUP.DISK

The **BACKUP.DISK** file attribute causes the output of the file to go to disk backup and not tape backup, if the PBD MCP option is set.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE LINE BACKUP.DISK;
```

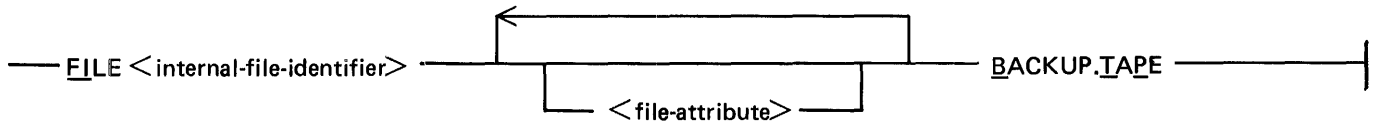
```
MODIFY PROG1; FILE LINE BDK;
```

BACKUP.TAPE

FILE BACKUP.TAPE

The BACKUP.TAPE file attribute causes the output of the file to go to tape backup and not disk backup, if the PBT MCP option is set.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

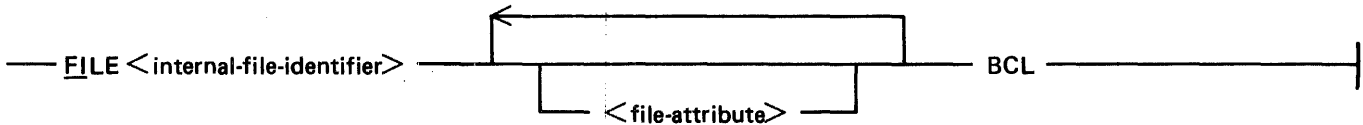
EXECUTE A/B; FILE LINE BACKUP.TAPE;

MODIFY PROG1; FILE LINE BTP;

BCL

The BCL file attribute specifies the recording mode to be the BCL recording mode. This attribute is only applicable for card and paper tape files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE MASTER BCL;

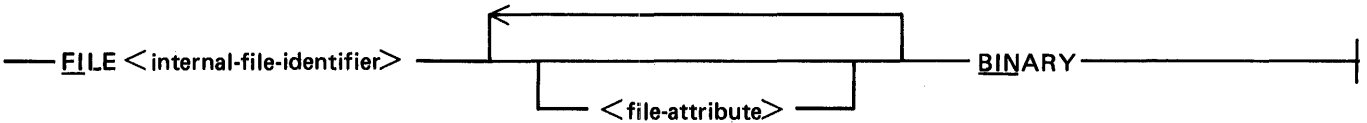
MODIFY PROG1; FILE PAYROLL BCL;

BINARY

FILE BINARY

The **BINARY** file attribute specifies the recording mode to be the binary recording mode and is used only for 80-column card and paper tape devices.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the **<file-attribute>** is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

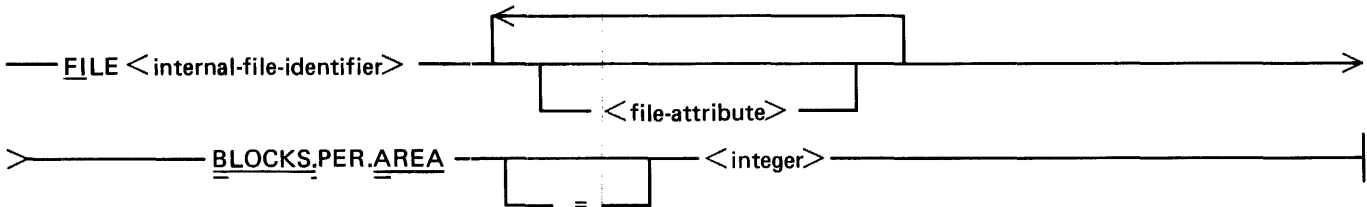
EXECUTE A/B; FILE MASTER BINARY;

MODIFY PROG1; FILE PAYROLL BIN;

BLOCKS.PER.AREA

The **BLOCKS.PER.AREA** file attribute assigns the number of physical records (blocks) to each disk area. This file attribute applies only to new files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 16,777,215 inclusive, and specifies the number of physical records (blocks) per area of disk for this file.

Examples:

```
EXECUTE A/B; FILE MASTER BLOCKS.PER.AREA = 30;
```

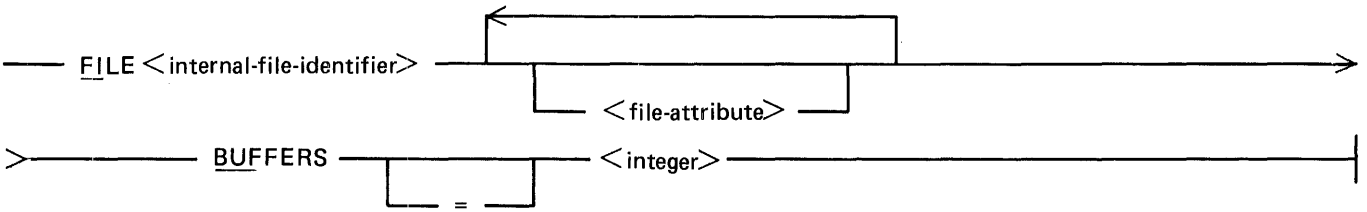
```
MODIFY PROG1; FILE PAYROLL B.A 40;
```

BUFFERS

FILE BUFFERS

The **BUFFERS** file attribute assigns the number of buffers for the file. If a value of zero is specified, the MCP assigns one buffer when the file is opened. For files with a device type of **QUEUE**, the **BUFFERS** file attribute specifies the maximum number of messages allowed in memory at any one time. For files with a device type of **REMOTE**, the **BUFFERS** file attribute specifies the maximum number of messages in the input queue allowed in memory at any one time.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 255 inclusive, and specifies the number of buffers to be used for the file.

Examples:

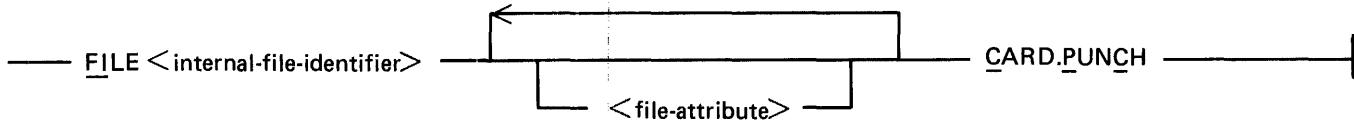
```
EXECUTE A/B; FILE MASTER BUFFERS = 3;
```

```
MODIFY PROG1; FILE QUEUE BUF 255;
```

CARD.PUNCH

The CARD.PUNCH file attribute specifies that the output file is to be sent to the card punch.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE PUNCH CARD.PUNCH;
```

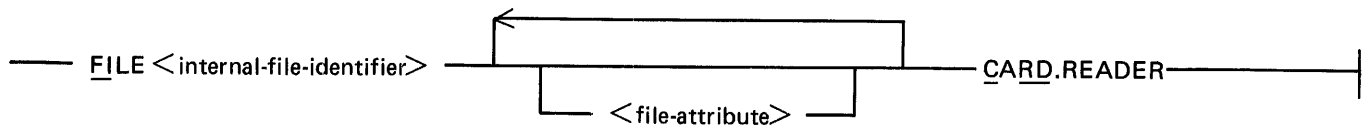
```
MODIFY PROG1; FILE OUTFILE CPC;
```

CARD.READER

FILE CARD.READER

The CARD.READER file attribute specifies that the input file is located on the card reader.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

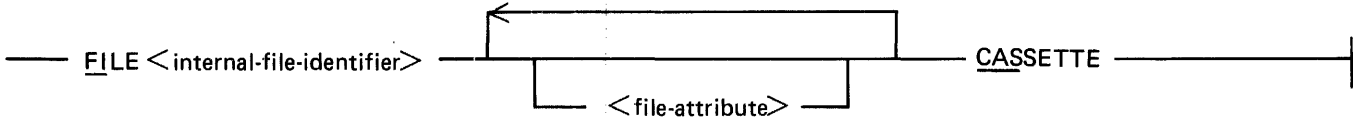
EXECUTE A/B; FILE INPUT CARD.READER;

MODIFY PROG1; FILE CARDIN CRD;

CASSETTE

The CASSETTE file attribute specifies that the file to be processed is located on a cassette.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE MASTER CASSETTE;

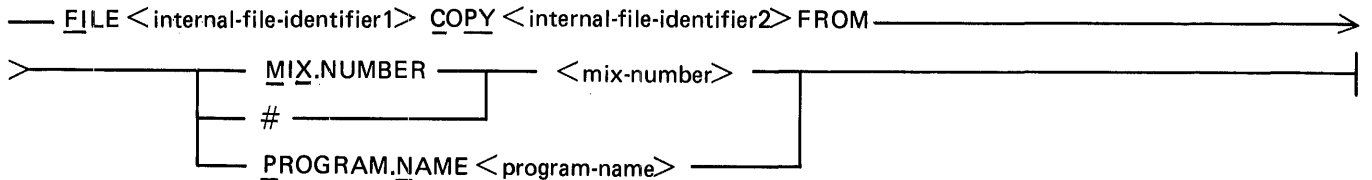
MODIFY PROG1; FILE CASFILE CAS;

COPY

FILE COPY

The COPY file attribute causes the File Parameter Block (FPB) of one file to be copied to the FPB of a receiving file. The internal file identifier of the receiving file is not changed.

Syntax:



Semantics:

internal-file-identifier1

This field can contain any valid internal file identifier of a program and specifies the file for which the FPB is being changed.

internal-file-identifier2

This field can contain any valid internal file identifier of a program and specifies the file from which the FPB is copied.

mix-number

This field can contain any mix number of an executing program and specifies the program containing the file from which the FPB is copied.

program-name

This field can contain any 30-character file name of a program that is currently executing on the system and specifies the program containing the file from which the FPB is copied.

Examples:

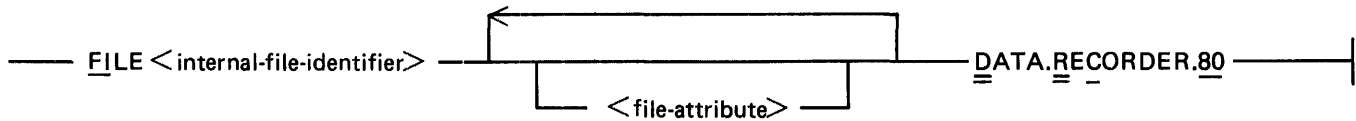
EXECUTE A/B; FILE MASTER COPY MASTERFILE FROM MIX.NUMBER 10;

MODIFY PROG1; FILE PAYROLL CPY PAYSAVE FROM PROGRAM.NAME PAYSAVE;

DATA.RECORDER.80

The DATA.RECORDER.80 file attribute specifies that the file is associated with the data recorder.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER DATA.RECORDER.80;
```

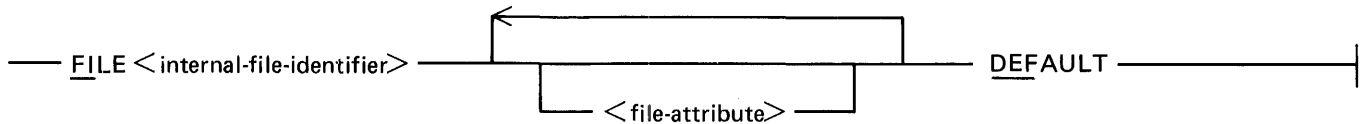
```
MODIFY PROG1; FILE OUTFILE DR80;
```


DEFAULT

FILE DEFAULT

The DEFAULT file attribute overrides the declared block and record sizes and uses the block and record sizes contained in the existing disk file header or tape label instead. This file attribute applies only to input disk files and labeled tape files created on B 7800/B 7700, B 6800/B 6700, and B 1000 systems. This file attribute applies only to existing files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

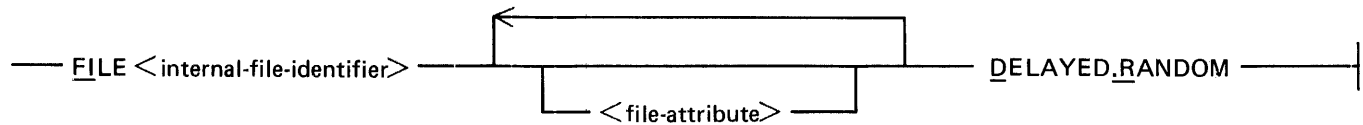
```
EXECUTE A/B; FILE MASTER DEFAULT;
```

```
MODIFY PROG1; FILE PAYROLL DEF;
```

DELAYED.RANDOM

The DELAYED.RANDOM file attribute specifies that the file-access method is delayed random. Refer to appendix C for a complete description of the delayed random, file-access method.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER DELAYED.RANDOM;
```

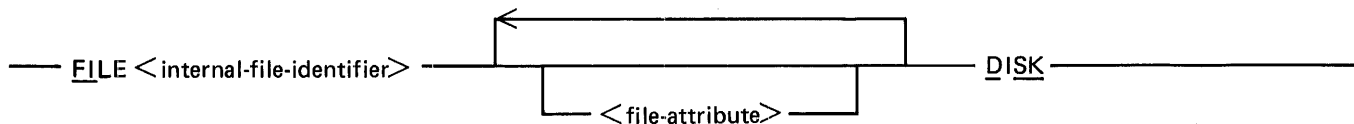
```
MODIFY PROG1; FILE PAYROLL D.R;
```

DISK

FILE DISK

The DISK file attribute specifies that the file is located on disk, regardless of the compiled program declaration.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

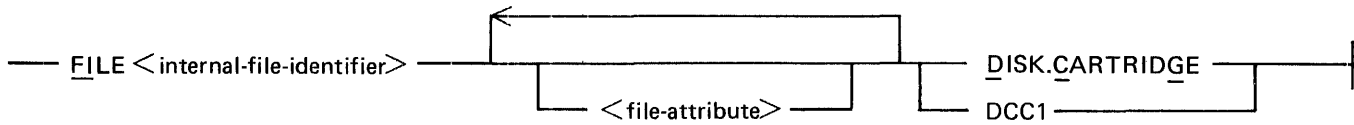
EXECUTE A/B; FILE MASTER DISK;

MODIFY PROG1; FILE INPUT DSK;

DISK.CARTRIDGE

The DISK.CARTRIDGE file attribute specifies that the file to be processed is located on a disk cartridge.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER DISK.CARTRIDGE;
```

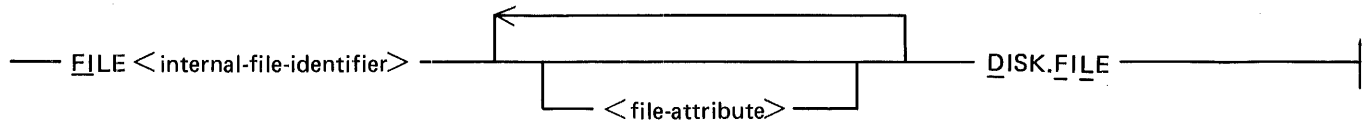
```
MODIFY PROG1; FILE INPUT DCC1;
```

DISK.FILE

FILE DISK.FILE

The DISK.FILE file attribute specifies that the file to be processed is located on disk.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

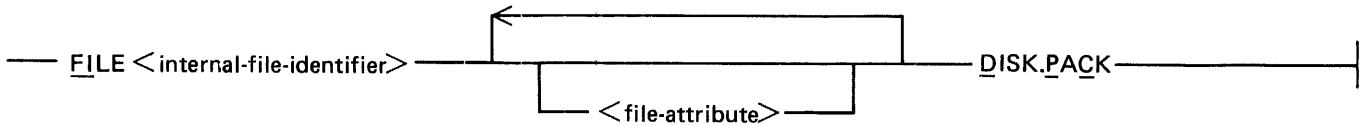
EXECUTE A/B; FILE MASTER DISK.FILE;

MODIFY PROG1; FILE INPUT DFL;

DISK.PACK

The DISK.PACK file attribute specifies that the file to be processed is located on a disk pack.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE MASTER DISK.PACK;

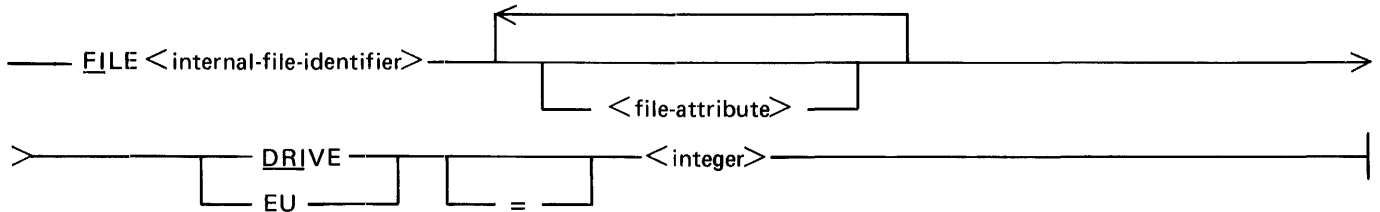
MODIFY PROG1; FILE INPUT DPC;

DRIVE

FILE DRIVE

The DRIVE or EU file attribute causes the file to be directed to the disk drive specified by <integer>. The disk drive must be a SYSTEM disk.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 15 inclusive, and specifies the drive number to which the file is directed.

Examples:

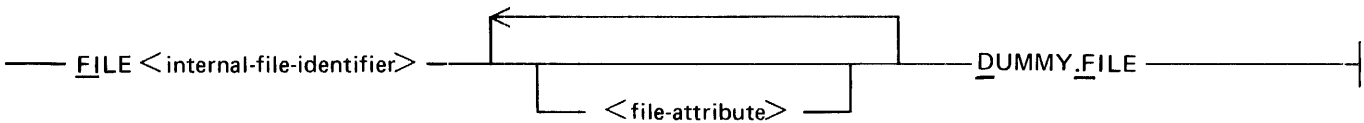
EXECUTE A/B; FILE MASTER DRIVE = 0;

MODIFY PROG1; FILE PAYROLL DRI 3;

DUMMY.FILE

The DUMMY.FILE file attribute causes the MCP to ignore any read or write operations for the file. A minimum File Information Block (FIB) is built with zero as the hardware type, and no buffers or input/output (I/O) descriptors are allocated. Upon performing an output operation to a dummy file, a program is reinstated with no data transfer or physical I/O taking place. The same is true for an input operation, unless the program specifies an end of file branch. In this case, the EOF branch is taken, the same as with a non-present optional file. Dummy files are particularly useful as debug files, or for cases in which the output of a program is not needed.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE DEBUGFILE DUMMY.FILE;
```

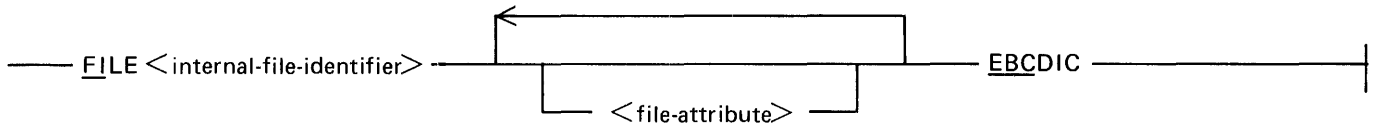
```
MODIFY PROG1; FILE ERRORS D.F;
```


EBCDIC

FILE EBCDIC

The EBCDIC file attribute causes the recording mode for the file to be the Extended Binary Coded Decimal Interchange Code (EBCDIC) recording mode.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

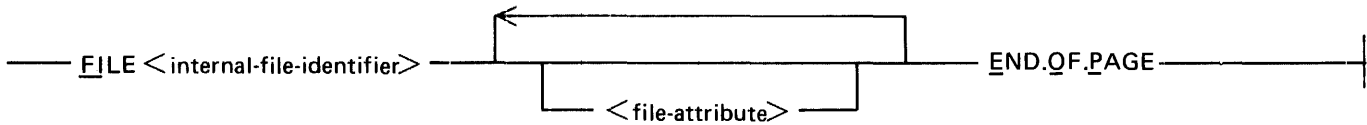
EXECUTE A/B; FILE MASTER EBCDIC;

MODIFY PROG1; FILE PAYROLL EBC;

END.OF.PAGE

The END.OF.PAGE file attribute causes the MCP to report the end of page to the program after sensing a channel 12 punch on a line printer file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE LINE END.OF.PAGE;
```

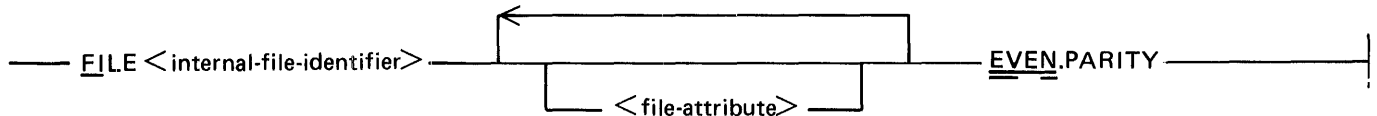
```
MODIFY PROG1; FILE LINE EOP;
```

EVEN.PARITY

FILE EVEN.PARITY

The EVEN file attribute causes the MCP to use even parity-checking for the file. This attribute is only applicable for tape files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

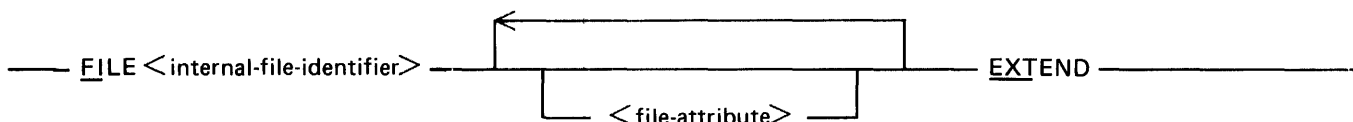
EXECUTE A/B; FILE MASTER EVEN.PARITY;

MODIFY PROG1; FILE PAYROLL EVEN;

EXTEND

The EXTEND file attribute applies to serial disk files that are opened with INPUT and OUTPUT and allows records to be added to the end of the file. The access characteristics of input/output (I/O) sequential files are also affected by the EXTEND file attribute. Refer to appendix C for a complete description of the sequential file access mechanism.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER EXTEND;
```

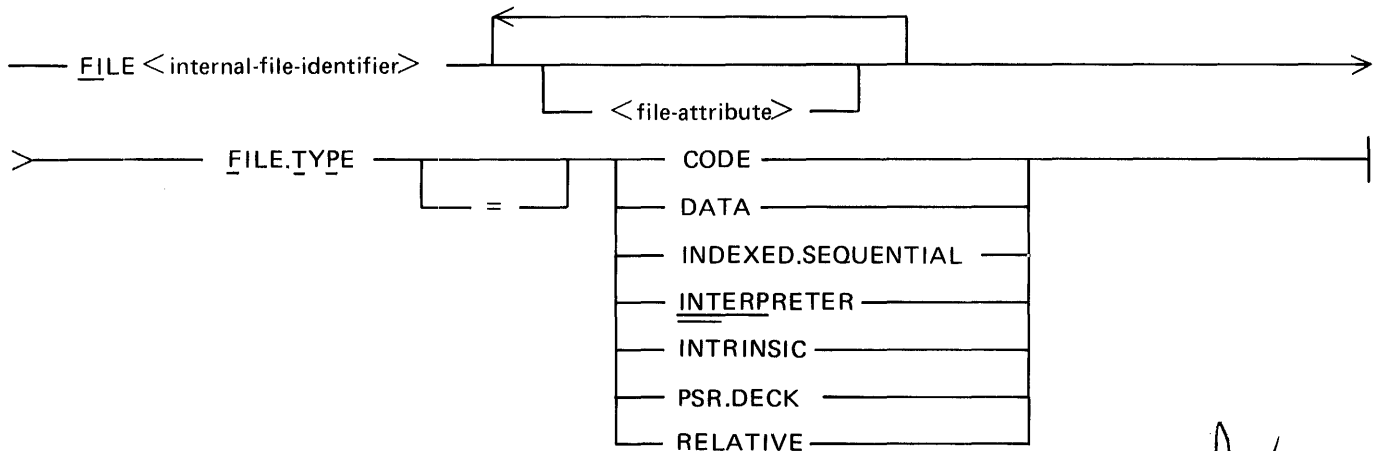
```
MODFIY PROG1; FILE PAYROLL EXT;
```

FILE.TYPE

FILE FILE.TYPE

The FILE.TYPE file attribute assigns a specific file type to an output disk file when it is closed and entered in the disk directory.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

CODE

The CODE keyword causes the file type to be an object code file that can be executed on the B 1000 system.

DATA

The DATA keyword causes the file type to be a data file.

INDEX.SEQUENTIAL

The INDEX.SEQUENTIAL keyword causes the file type to be an indexed sequential file used by COBOL74 programs.

INTERPRETER

The INTERPRETER keyword causes the file type to be an interpreter file.

INTRINSIC

The INTRINSIC keyword causes the file type to be an intrinsic file.

PSR.DECK

The PSR.DECK keyword causes the file type to be a psuedo-reader file read by the SYSTEM/LDCONTRL program using the LD system command described in section 5.

RELATIVE

The RELATIVE keyword causes the file type to be a relative file used by COBOL74 programs.

Examples:

EXECUTE A/B; FILE MASTER FILE.TYPE DATA;

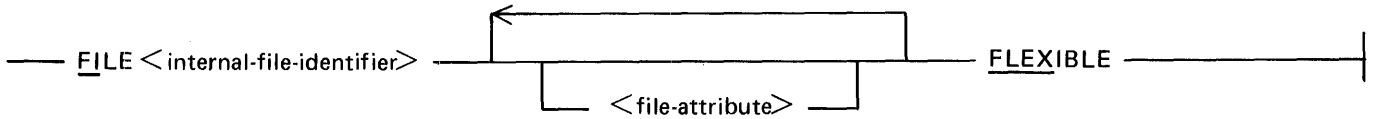
MODIFY PROG1; FILE TESTCODE FTP CODE;

FLEXIBLE

FILE FLEXIBLE

The FLEXIBLE file attribute allows the specified value of the AREAS file attribute to be exceeded, up to the system limit of 105 areas. The FLEXIBLE file attribute applies only to disk files and can be changed only when the file is closed. This attribute does not work for ISAM files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

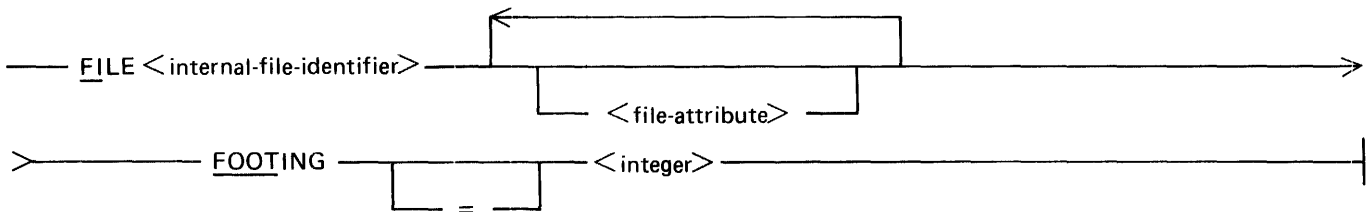
EXECUTE A/B; FILE OUTPUT FLEXIBLE;

MODIFY PROG1; FILE DISK FLEX;

FOOTING

The FOOTING file attribute specifies the number of lines from the beginning of the page body to the line where the MCP begins to report an end-of-page condition to the program. When this condition occurs, the ON EOF branch of the program is taken, but the record is still written, unless the record is written in the lower margin, in which case a page advance is made to the beginning of the next logical page.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 255 inclusive, and specifies the number of lines from the beginning of the page body to the line where the MCP begins to report an end-of-page condition.

Examples:

```
EXECUTE A/B; FILE PRINT FOOTING = 50;
```

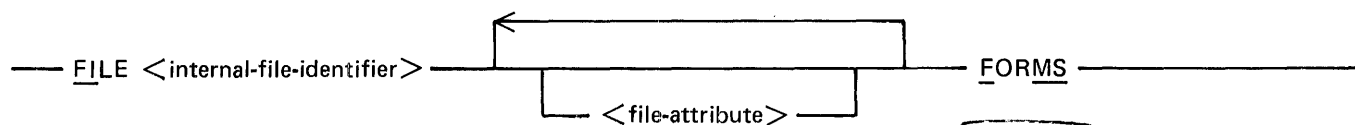
```
MODIFY PROG1; FILE CHECKS FOOT 10;
```


FORMS

FILE FORMS

The FORMS file attribute causes the program to be suspended and causes the MCP to display a message requesting the operator to load special forms in a device (line printer or card punch) before the file is opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

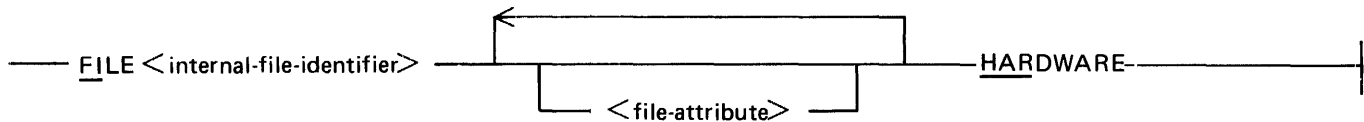
EXECUTE A/B; FILE LINE FORMS;

MODIFY PROG1; FILE CHECKS FMS;

HARDWARE

The **HARDWARE** file attribute allows a printer or punch file to go to the hardware device assigned.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE LINE HARDWARE;

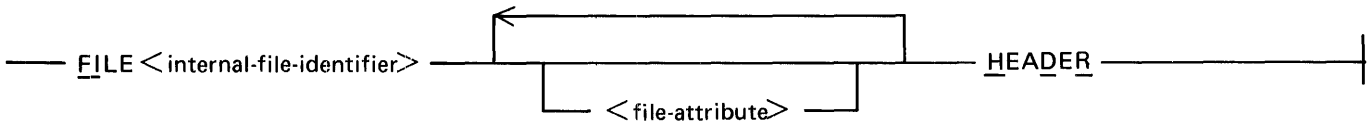
MODIFY PROG1; FILE PRINTER HAR;

HEADER

FILE HEADER

The **HEADER** file attribute applies to files with a device type of **REMOTE** and causes a 50-byte message header to be included in remote file read and write operations.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the **<file-attribute>** is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

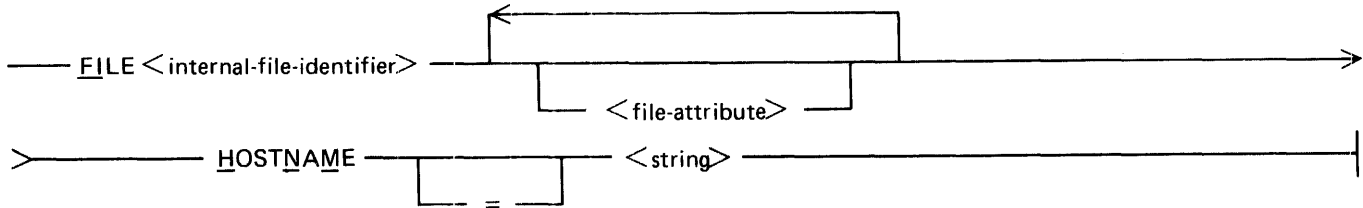
EXECUTE MCS; FILE REMOTE HEADER;

EXECUTE MCS; FILE MCSQUEUE HDR;

HOSTNAME

The **HOSTNAME** file attribute specifies a string that serves as the identifier of the host system where the file is to be located.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the **<file-attribute>** is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

string

This field can contain any 17-character string that specifies the name of the host system where the file is to be located.

Examples:

```
EXECUTE A/B; FILE MASTER HOSTNAME = B1000HOST;
```

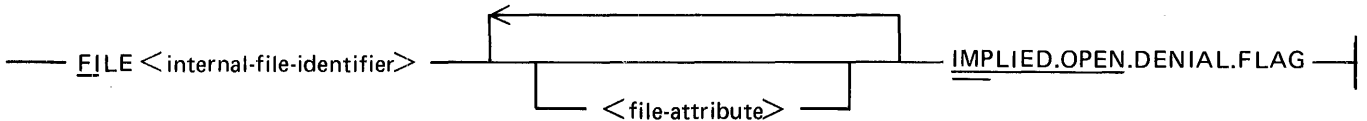
```
MODIFY PROG1; FILE PAYROLL HNM B6800HOST;
```

IMPLIED.OPEN.DENIAL.FLAG

FILE IMPLIED.OPEN.DENIAL.FLAG

The IMPLIED.OPEN.DENIAL.FLAG file attribute prevents the file from being opened implicitly. An implied open of a file occurs when a read or write operation is performed and the file was not explicitly opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

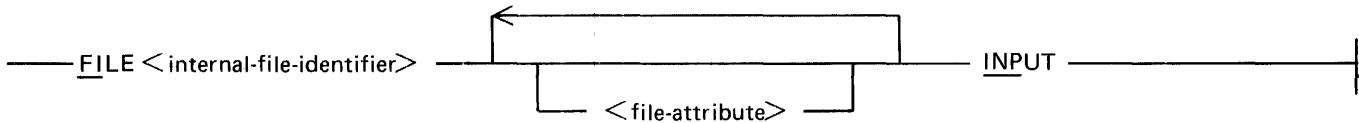
```
EXECUTE A/B; FILE MASTER IMPLIED.OPEN;
```

```
MODIFY PROG1; FILE PAYROLL NO IMP;
```

INPUT

The INPUT file attribute applies to files having an implied open. The INPUT file attribute sets the INPUT attribute and does not affect the settings of other implied open attributes. Implied open attributes are ignored by the MCP if the file is explicitly opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the < file-attribute > is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER INPUT;
```

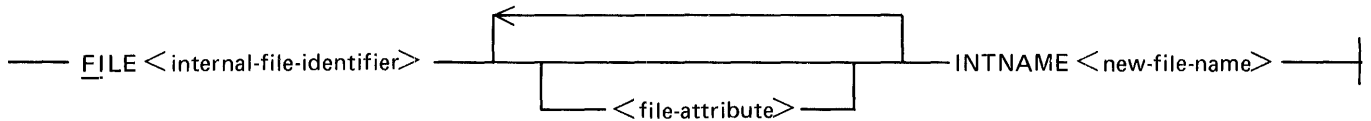
```
MODIFY PROG1; FILE PAYROLL INP;
```

INTNAME

FILE INTNAME

The INTNAME file attribute is used to modify the internal file name of a file. It can only be used with the MODIFY program control instruction.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

new-file-name

This field can contain any valid internal file identifier and specifies the new file name for the file.

Examples:

```
MODIFY A/B; FILE PRINT INTNAME LINE;
```

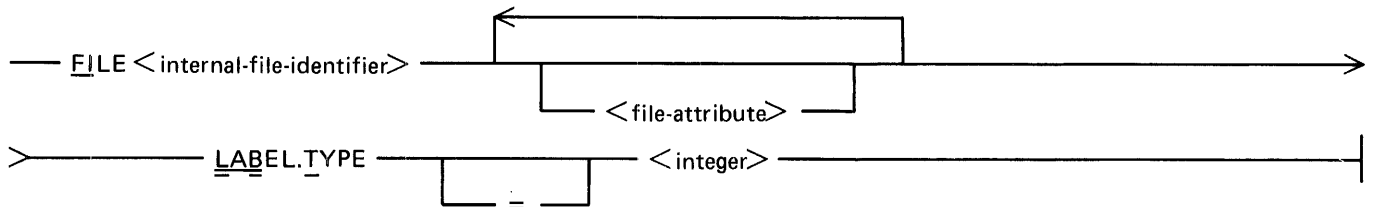
```
MO PROG1; FILE FILE3 INTNAME PAYROLL
```


LABEL.TYPE

FILE LABEL.TYPE

The LABEL.TYPE file attribute specifies which label type to use or to expect for the file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 2 inclusive, and specifies the type of label to use or expect for the file.

If the value of <integer> is 0, the American National Standard Institute (ANSI) label is used or expected.

If the value of <integer> is 1, the file is unlabeled.

If the value of <integer> is 2, the Burroughs standard label is used or expected.

Examples:

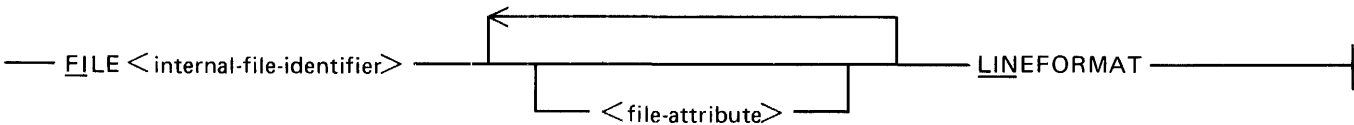
```
EXECUTE A/B; FILE LINE LABEL.TYPE = 2;
```

```
MODIFY PROG1; FILE TAPE LAB 0;
```

LINEFORMAT

The LINEFORMAT file attribute causes the values set for the FOOTING, LOWER.MARGIN, PAGE.SIZE, and UPPER.MARGIN file attributes to be used for printer files. Specifying NOT LINEFORMAT causes the FOOTING, LOWER.MARGIN, PAGE.SIZE, and UPPER.MARGIN file attributes to be ignored when printing.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE PRINT LINEFORMAT;

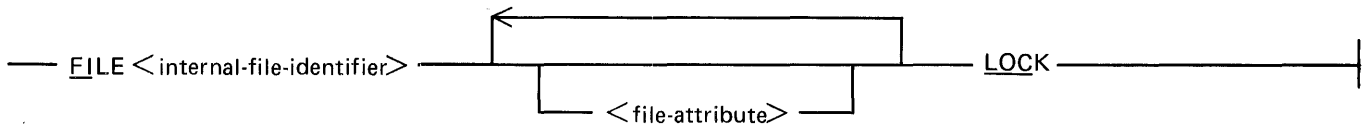
MODIFY PROG1; FILE CHECKS NO LIN;

LOCK

FILE LOCK

The LOCK file attribute causes a disk file to be locked in the disk directory if the file is still open when the program goes to end of job or the program is discontinued with the DS or DP system commands described in section 5. This file attribute applies only to new files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

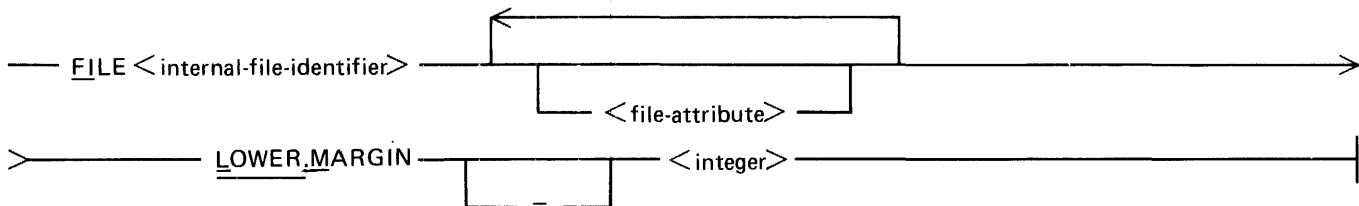
EXECUTE A/B; FILE MASTER LOCK;

MODIFY PROG1; FILE DISK LOC;

LOWER.MARGIN

The LOWER.MARGIN file attribute specifies the number of lines to leave blank between the page body and the bottom of the form.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 255 inclusive, and specifies the number of lines to leave blank between the page body and the bottom of the form.

Examples:

```
EXECUTE A/B; FILE PRINT LOWER.MARGIN = 6;
```

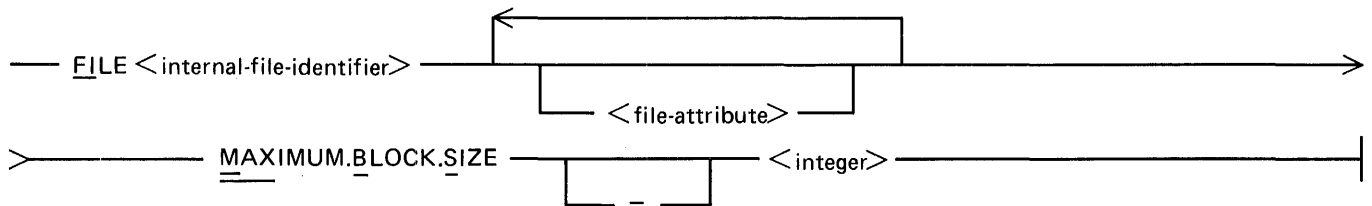
```
MODIFY PROG1; FILE CHECKS L.M 3;
```

MAXIMUM.BLOCK.SIZE

FILE MAXIMUM.BLOCK.SIZE

The MAXIMUM.BLOCK.SIZE file attribute specifies the largest block size to be used for variable-length records.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any valid integer in the range 1 to 16,777,215 inclusive, and specifies the maximum number of bits for the block size to be used for variable-length records.

Examples:

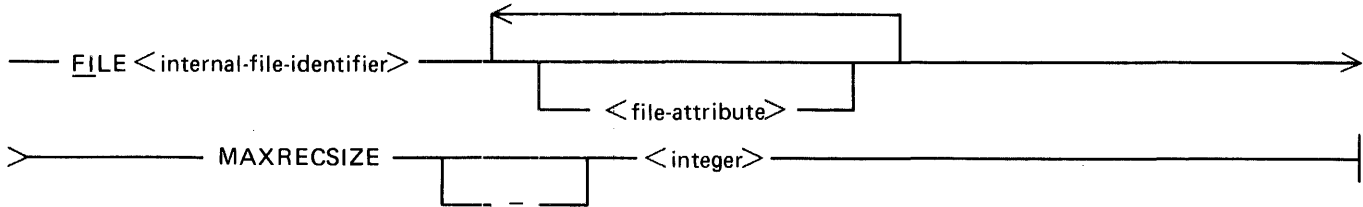
EXECUTE A/B; FILE MASTER MAXIMUM.BLOCK.SIZE = 1800;

MODIFY PROG1; FILE DISK MAX 2000;

MAXRECSIZE

The MAXRECSIZE file attribute specifies the maximum record size in bits for variable-length record files. This file attribute is equivalent to the RECORD.SIZE.IN.BYTES file attribute.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 16,777,215 inclusive, and specifies the maximum record size in bits for a variable-length record file.

Examples:

```
EXECUTE A/B; FILE VARFILE MAXRECSIZE = 1440;
```

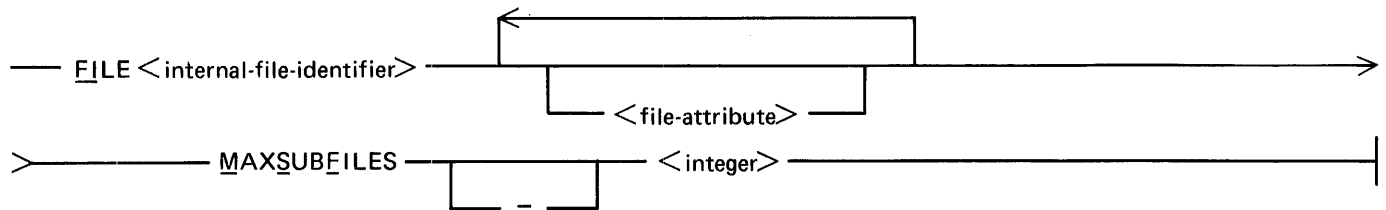
```
MODIFY PROG1; FILE INPUT MAXRECSIZE 14400;
```

MAXSUBFILES

FILE MAXSUBFILES

The MAXSUBFILES file attribute specifies the maximum number of subfiles in a file with a device type of PORT.FILE.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 255 inclusive, and specifies the maximum number of subfiles for the PORT file.

Examples:

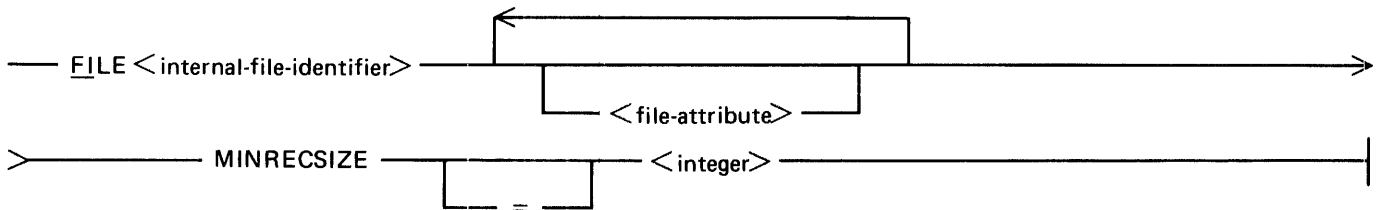
```
EXECUTE A/B; FILE PORTIO MAXSUBFILES = 10;
```

```
MODIFY PROG1; FILE PAYROLL MSF 5;
```

MINRECSIZE

The MINRECSIZE file attribute specifies the minimum record size in bits for variable-length record files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 16,777,215 inclusive, and specifies the minimum record size in bits for the variable-length record file.

Examples:

```
EXECUTE A/B; FILE IN MINRECSIZE = 1440;
```

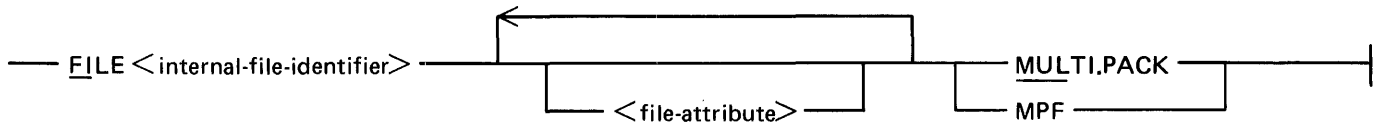
```
MODIFY PROG1; FILE PAYROLL MINRECSIZE 14400;
```


MULTI.PACK

FILE MULTI.PACK

The MULTI.PACK file attribute ^{allows} ~~causes~~ the disk file to use more than one user disk.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

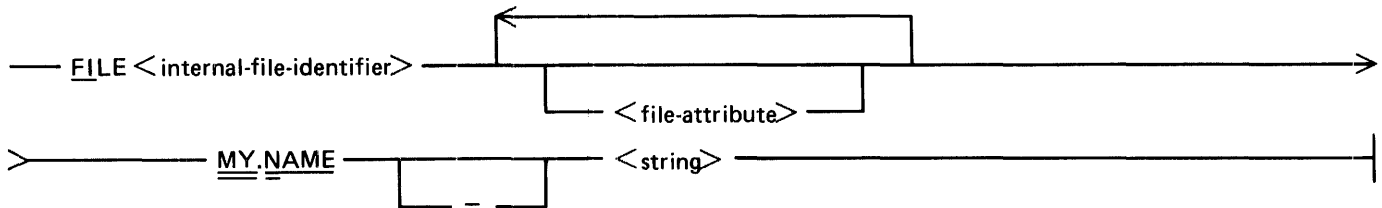
EXECUTE A/B; FILE MASTER MULTI.PACK;

MODIFY PROG1; FILE DISK MPF;

MY.NAME

The MY.NAME file attribute applies to files with a device type of PORT.FILE and specifies the identification by which the program opening the port file wishes to be known.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

string

This field can contain any 17 characters and specifies the name by which the program opening the port file wishes to be known.

Examples:

```
EXECUTE A/B; FILE PORTIO MY.NAME = B1000HOST;
```

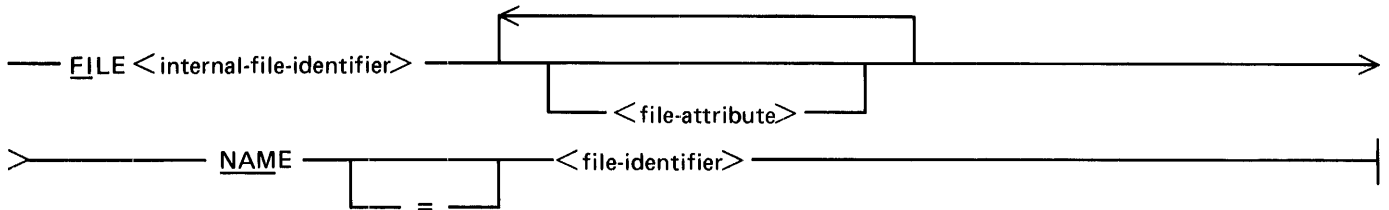
```
MODIFY PROG1; FILE PAYROLL MYN PAYROLLHST;
```

NAME

FILE NAME

The NAME file attribute causes the external file identifier or family name (disk identifier) to be changed to the value of <file-identifier>. If only the family name is to be changed, use the PACK.ID file attribute instead.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

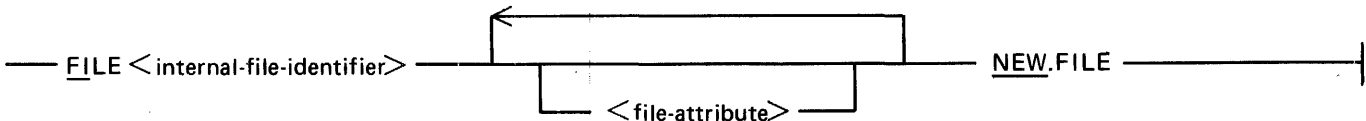
EXECUTE A/B; FILE MASTER NAME MASTERPACK/MASTER/FILE;

MODIFY PROG1; FILE DISK NAM PROG1/DISKFILE;

NEW.FILE

The NEW.FILE file attribute applies to implied opens of a file and sets the NEW attribute and does not affect the setting of the other implied open attributes. An implied open of a file occurs when the program performs a read or write operation on a file and the program did not perform an open operation. The implied open attributes are ignored if the file is explicitly opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER NEW.FILE;
```

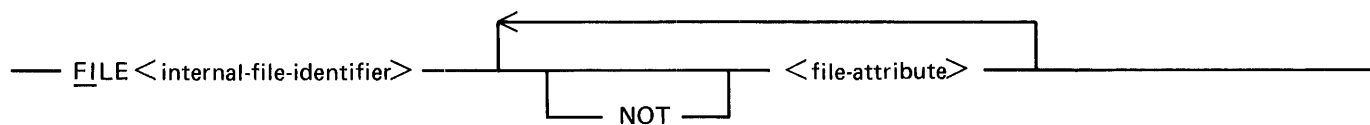
```
MODIFY PROG1; FILE LINE NEW;
```

NOT

FILE NOT

The NOT file attribute negates the immediately following file attribute. For example, a file assigned to go only to backup could be changed to go to the line printer by specifying NO BACKUP in the FILE program control instruction.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

file-attribute

This field can contain any of the following file attributes and specifies the file attribute to negate.

ALLOCATE.AT.OPEN	NEW.FILE
AUTOPRINT	OPEN.LOCK
BACKUP	OPEN.LOCKOUT
BACKUP.DISK	OPTIONAL
BACKUP.TAPE	OUTPUT
DEFAULT	PSEUDO
DUMMY.FILE	REVERSE
END.OF.PAGE	REWIND
EXTEND	SPECIAL.FORMS
HARDWARE	TRANSLATE
HEADER	USER.BACKUP.NAME
IMPLIED.OPEN	WITH.INTERPRET
INPUT	WITH.PRINT
LOCK	WITH.STACKERS
MULTI.PACK	WORK.FILE

Examples:

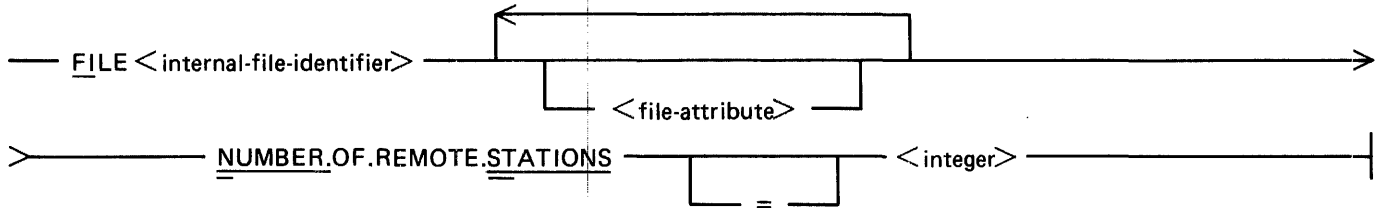
EXECUTE A/B; FILE LINE NOT HARDWARE;

MODIFY PROG1; FILE TAPE NO REWIND;

NUMBER.OF.REMOTE.STATIONS

The NUMBER.OF.REMOTE.STATIONS file attribute applies to files with a device type of REMOTE and specifies the maximum number of stations that can be attached to the file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 999 inclusive, and specifies the maximum number of stations that can be attached to this remote file.

Examples:

```
EXECUTE MCS; FILE REMOTE NUMBER.OF.REMOTE.STATIONS = 12;
```

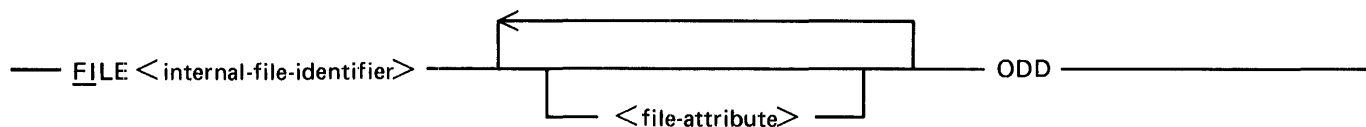
```
EXECUTE PROG1; FILE MCSQUEUE NUMBER.STATIONS 2;
```

ODD

FILE ODD

The ODD file attribute causes the MCP to use odd parity checking for this file. This attribute is applicable to magnetic tape files only.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE MASTER ODD;

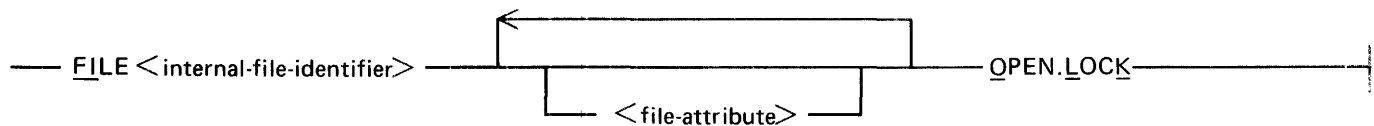
MODIFY PROG1; FILE TAPE ODD;

OPEN.LOCK

The OPEN.LOCK file attribute sets the LOCK attribute when the file is opened implicitly. An implied file open occurs when a program performs a read or write operation on a file without explicitly opening the file. If the file is explicitly opened, then the implied open attributes are ignored.

The OPEN.LOCK attribute gives a program the ability to prevent other programs from writing to the file while the first program has the file open. Other programs are not prevented from reading the file while the first program has the file open.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER OPEN.LOCK;
```

```
MODIFY PROG1; FILE DISK OLK;
```

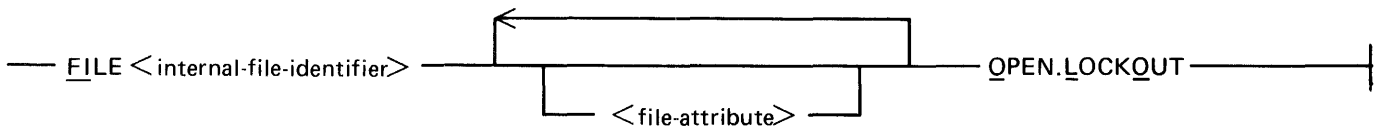

OPEN.LOCKOUT

FILE OPEN.LOCKOUT

The OPEN.LOCKOUT file attribute sets the LOCKOUT attribute when the file is opened implicitly. An implied file open occurs when a program performs a read or write operation to a file without explicitly opening the file. If the file is explicitly opened, then the implied open attributes are ignored.

The OPEN.LOCKOUT file attribute prevents other programs from accessing the file while the first program has the file open. The first program that opens the file with the OPEN.LOCKOUT file attribute set has exclusive use of the file.

Syntax:



Examples:

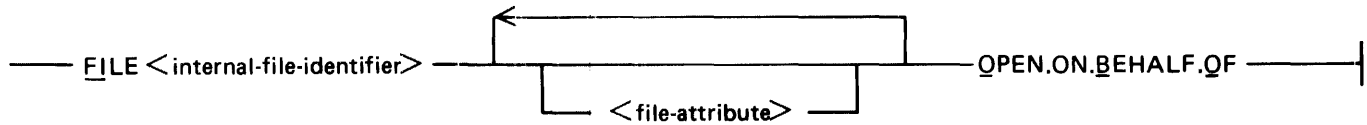
EXECUTE A/B; FILE MASTER OPEN.LOCKOUT;

MODIFY PROG1; FILE DISK OLO;

OPEN.ON.BEHALF.OF

The OPEN.ON.BEHALF.OF file attribute causes the security restrictions associated with the usercode/password of the task to not be applied to the file. The OPEN.ON.BEHALF.OF file attribute can only be changed when the file is closed.

Syntax:



Examples:

EXECUTE A/B; FILE MASTER OPEN.ON.BEHALF.OF;

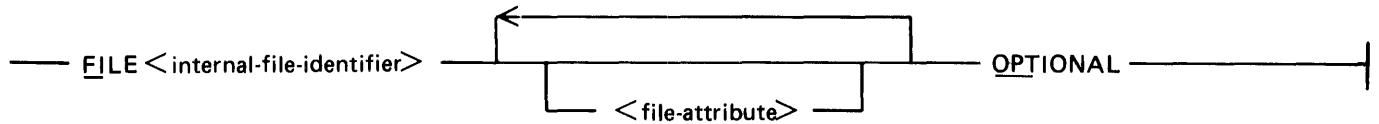
MODIFY PROG1; FILE DISK OBO;

OPTIONAL

FILE OPTIONAL

The OPTIONAL file attribute causes the file to be an optional file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

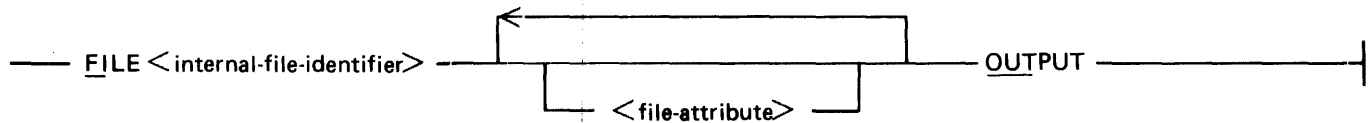
EXECUTE A/B; FILE DEBUG OPTIONAL;

MODIFY PROG1; FILE TEST OPT;

OUTPUT

The OUTPUT file attribute sets the OUTPUT attribute for a file opened implicitly. An implied open of a file occurs when a program performs a write operation to a file and an open operation was not performed. If the file is opened explicitly, then the implied open attributes are ignored.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE LINE OUTPUT;

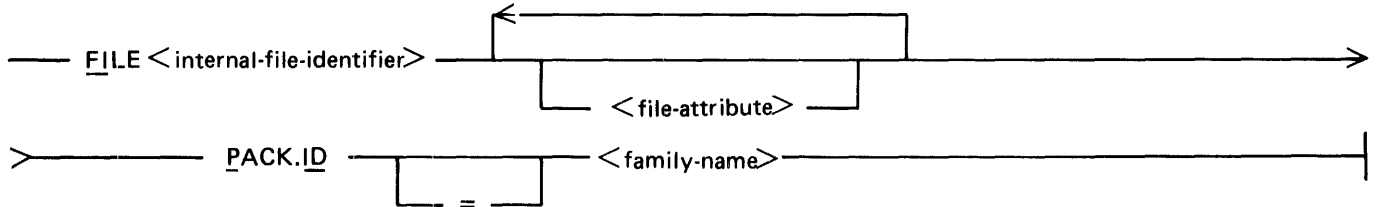
MODIFY PROG1; FILE TAPE OUT;

FILE PACK.ID

PACK.ID

The PACK.ID file attribute assigns a disk identifier to the file. A subsequent NAME or TITLE file attribute overrides the value assigned to the disk identifier.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

disk-identifier

This field can contain any 10-character disk identifier that is valid on the B 1000 system.

Examples:

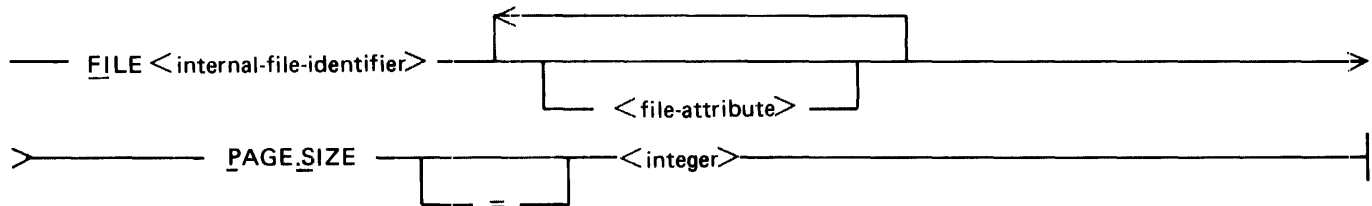
EXECUTE A/B; FILE MASTER PACK.ID = USER;

MODIFY PROG1; FILE PAYROLL PID FINANCE;

PAGE.SIZE

The PAGE.SIZE file attribute specifies the number of lines between the lower and upper margins of a printed form. The lower margin is specified with the LOWER.MARGIN file attribute and the upper margin is specified with the UPPER.MARGIN file attribute.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 255 inclusive, and specifies the number of lines between the lower and upper margins of a printed form.

Examples:

```
EXECUTE A/B; FILE FORMS PAGE.SIZE = 30;
```

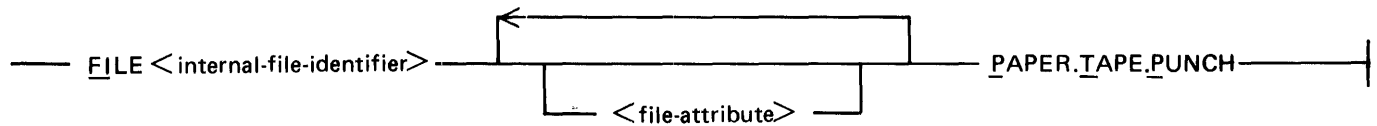
```
MODIFY PROG1; FILE CHECKS P.S 20;
```

FILE PAPER.TAPE.PUNCH

PAPER.TAPE.PUNCH

The PAPER.TAPE.PUNCH file attribute specifies that output is to be a paper-tape-punch device.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

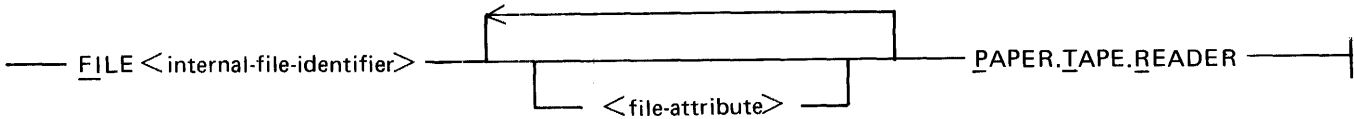
EXECUTE A/B; FILE MASTER PAPER.TAPE.PUNCH;

MODIFY PROG1; FILE OUTPUT PTP;

PAPER.TAPE.READER

The PAPER.TAPE.READER file attribute specifies that input is from a paper-tape-reader device.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER PAPER.TAPE.READER;
```

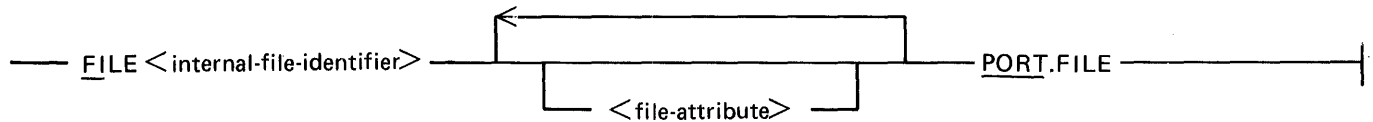
```
MODIFY PROG1; FILE OUTPUT PTR;
```


FILE PORT.FILE

PORT.FILE

The PORT.FILE file attribute specifies that the device for the file is to be of type PORT.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

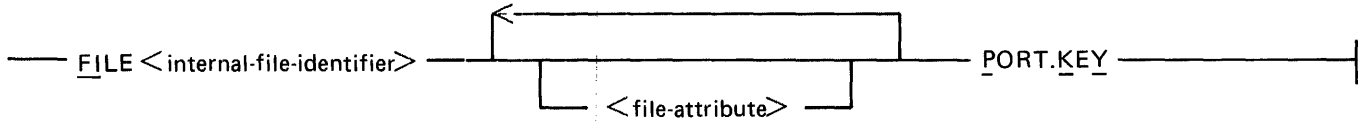
EXECUTE A/B; FILE PORTIO PORT.FILE;

MODIFY PROG1; FILE PAYPORT PORT;

PORT.KEY

The PORT.KEY file attribute specifies that the port file has keys (subport indexes) used in the read and write operations.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

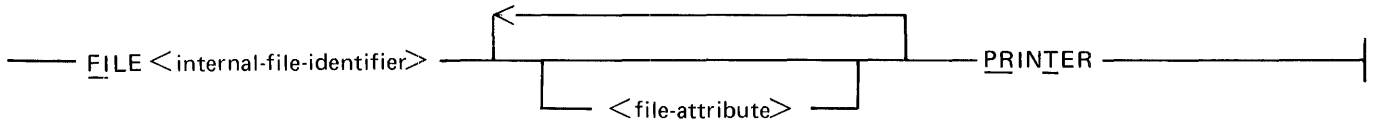
```
EXECUTE A/B; FILE PORTIO PORT.KEY;
```

```
MODIFY PROG1; FILE PAYPORT PKY;
```

PRINTER

The PRINTER file attribute specifies that the output file is to be sent to an available printer.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute>, is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

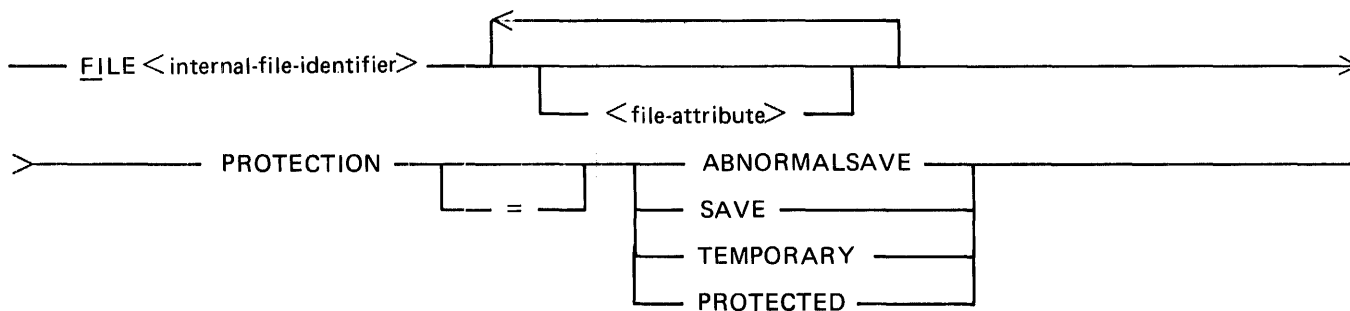
EXECUTE A/B; FILE OUTPUT PRINTER;

MODIFY PROG1; FILE OUTFILE PRT;

PROTECTION

The PROTECTION file attribute specifies additional protection for disk files in cases where the program is discontinued with the DS or DP system commands described in section 5. This file attribute applies only to new files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

ABNORMALSAVE

The ABNORMALSAVE keyword causes the disk file to be entered in the disk directory if the program that opened this file is discontinued with a DS or DP system command.

SAVE

The SAVE keyword causes the disk file to be entered in the disk directory as soon as the file is opened.

TEMPORARY

The TEMPORARY keyword causes a new disk file to be discarded when the program is discontinued with a DS or DP system command, unless the disk file has been explicitly closed.

PROTECTED

The PROTECTED keyword provides protection from loss of data from the file in the event of a system halt or program failure. See appendix C of this manual for a complete description of the effects of PROTECTION = PROTECTED.

Examples:

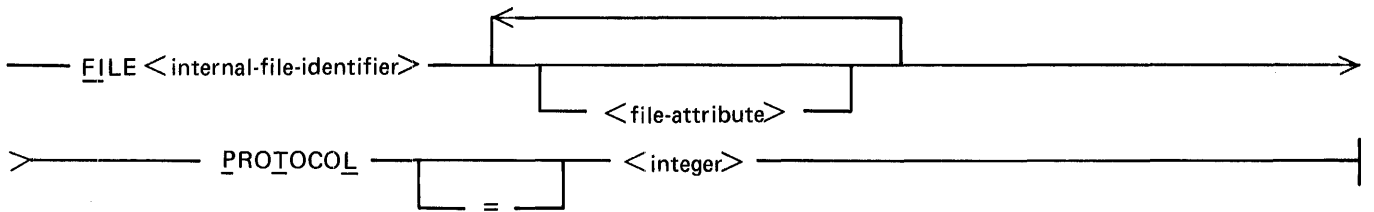
```
EXECUTE A/B; FILE MASTER PROTECTION = ABNORMALSAVE;
```

```
MODIFY PROG1; FILE PAYROLL PROTECTION SAVE;
```

PROTOCOL

The PROTOCOL file attribute applies only to files with device type of REMOTE and specifies the protocol number that is placed into the OPEN (type 10) message passed to a Message Control System (MCS) when the remote file is opened. Refer to the B 1000 Systems Network Definition Language (NDL) Language Manual for a complete description of the OPEN message. The PROTOCOL file attribute can be used for any desired purpose to communicate a value to an MCS at remote-file open time; for example, to define a specific communication protocol.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 99 inclusive, and specifies the protocol value to be placed in the open message passed to an MCS when the remote file is opened.

Examples:

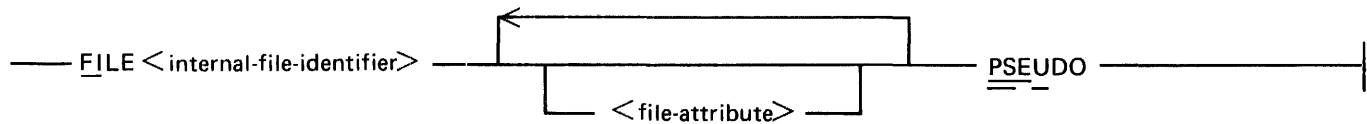
EXECUTE MCS; FILE REMOTE PROTOCOL = 10;

MODIFY PROG1; FILE MCSQUEUE PTL 2;

PSEUDO

The PSEUDO file attribute designates the file to have a PSEUDO READER disk file type.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

EXECUTE A/B; FILE OUT PSEUDO;

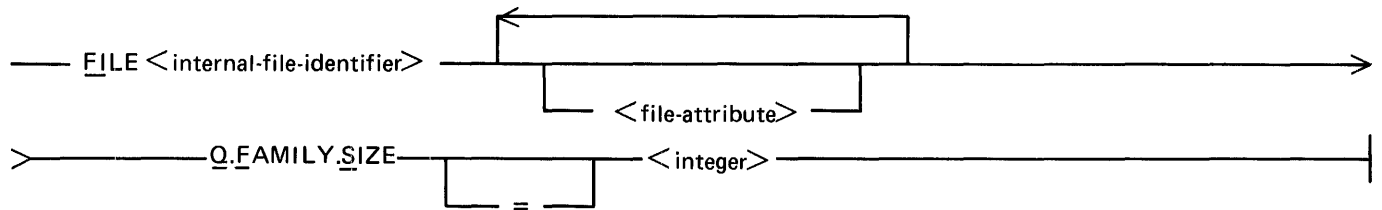
MODIFY PROG1; FILE PSEUDOFIELD PSU;

FILE Q.FAMILY.SIZE

Q.FAMILY.SIZE

The Q.FAMILY.SIZE file attribute applies only to files with a device type equal to QUEUE and specifies the number of subqueues in the file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to ¹⁰⁰⁰~~1000~~ inclusive, and specifies the number of subqueues in the queue-file first for this file.

Examples:

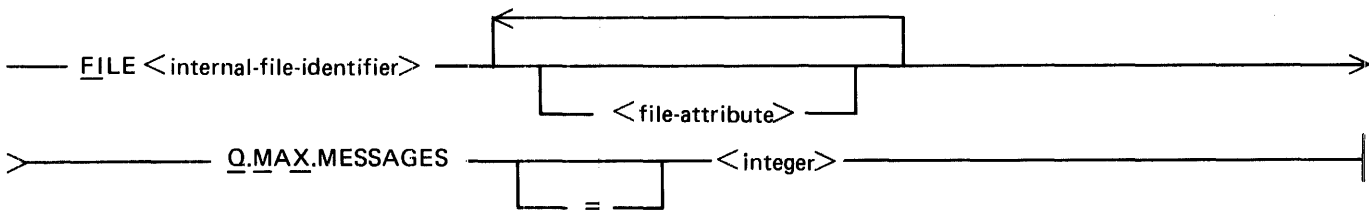
EXECUTE A/B; FILE MCSQUEUE Q.FAMILY.SIZE = 10;

MODIFY PROG1; FILE Q QFS 3;

Q.MAX.MESSAGES

The Q.MAX.MESSAGES file attribute applies to files with device types of QUEUE or REMOTE. For queue files, the Q.MAX.MESSAGES file attribute specifies the maximum number of messages that can be in the queue at any one time. For remote files, the Q.MAX.MESSAGES file attribute specifies the maximum number of messages that can be in the input queue. The size of output queue for remote files is declared by the network controller.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 1023 inclusive, and specifies the maximum number of messages that can be in the queue at any one time.

Examples:

```
EXECUTE A/B; FILE QUEUE Q.MAX.MESSAGES = 255;
```

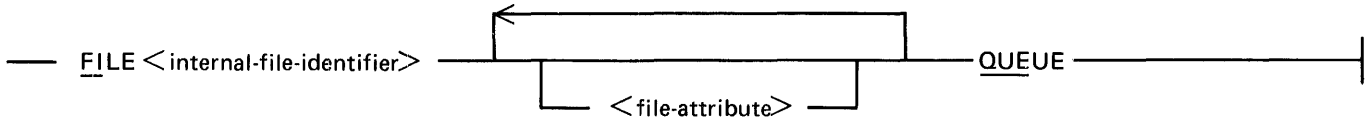
```
MODIFY PROG1; FILE REMOTE QMX 20;
```


QUEUE

FILE QUEUE

The QUEUE file attribute identifies the file as a queue file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

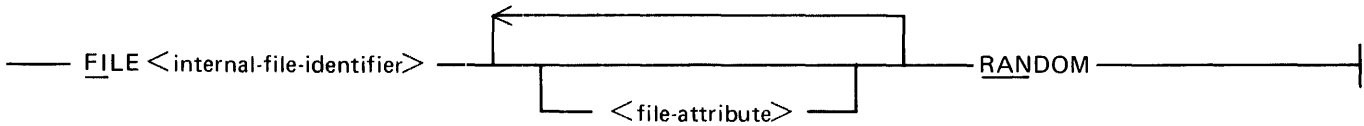
EXECUTE A/B; FILE MASTER QUEUE;

MODIFY PROG1; FILE QFILE QUE;

RANDOM

The RANDOM file attribute causes the access mode for the file to be random.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER RANDOM;
```

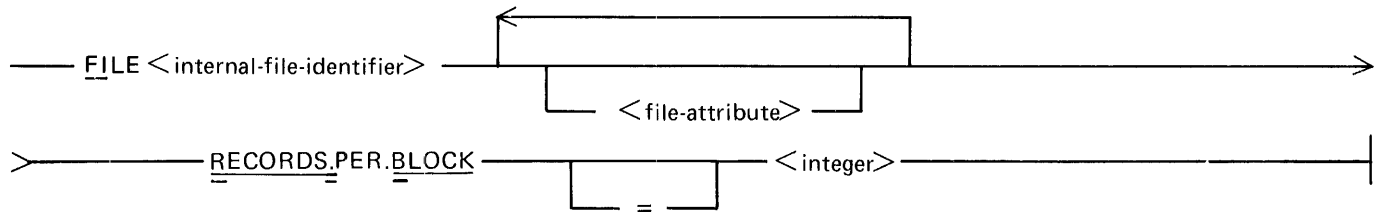
```
MODIFY PROG1; FILE DISK RAN;
```

FILE RECORDS.PER.BLOCK

RECORDS.PER.BLOCK

The RECORDS.PER.BLOCK file attribute specifies the number of logical records assigned per block for a fixed-length record file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 16,777,215 inclusive, and specifies the number of fixed-length records contained in each block.

Examples:

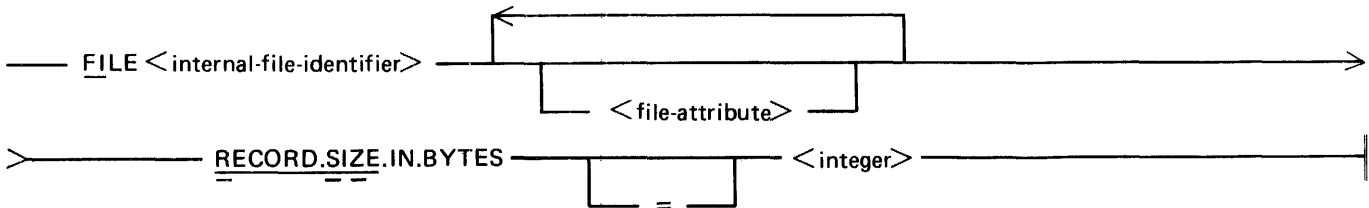
EXECUTE A/B; FILE MASTER RECORDS.PER.BLOCK = 10;

MODIFY PROG1; FILE DISK R.B 20;

RECORD.SIZE.IN.BYTES

The RECORD.SIZE.IN.BYTES file attribute assigns the number of bytes for each logical record in the file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 16,777,215 inclusive, and specifies the number of bytes for each logical record in the file.

Examples:

```
EXECUTE A/B; FILE MASTER RECORD.SIZE.IN.BYTES = 180;
```

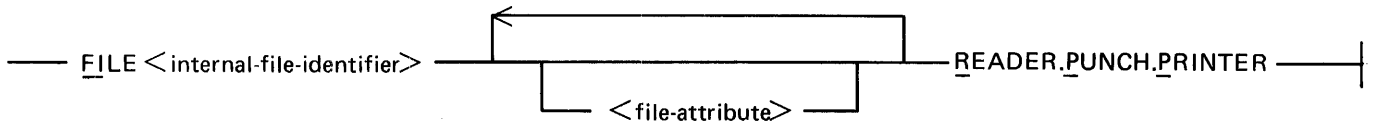
```
MODIFY PROG1; FILE PAYROLL RSZ 90;
```

FILE READER.PUNCH.PRINTER

READER.PUNCH.PRINTER

The READER.PUNCH.PRINTER file attribute assigns an input or output file to a card reader/punch device.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

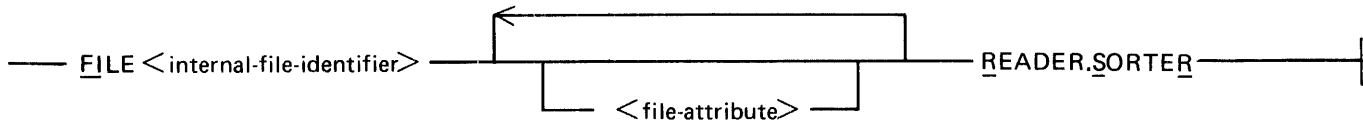
EXECUTE A/B; FILE MASTER READER.PUNCH.PRINTER;

MODIFY PROG1; FILE INPUT RPP;

READER.SORTER

The `READER.SORTER` file attribute assigns an input file to the reader-sorter device.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER READER.SORTER;
```

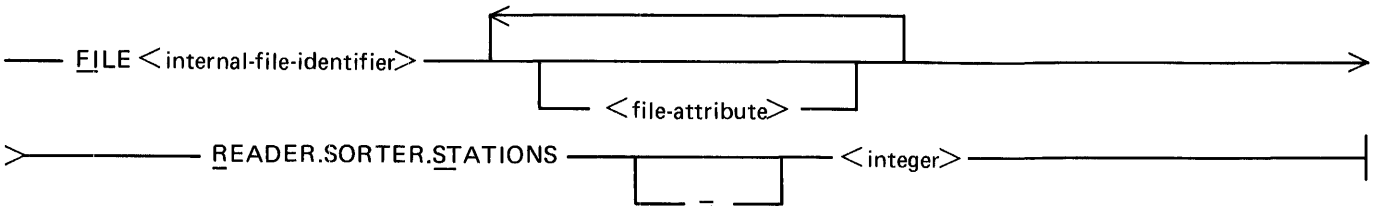
```
MODIFY PROG1; FILE INPUT RSR;
```

FILE READER.SORTER.STATIONS

READER.SORTER.STATIONS

The READER.SORTER.STATIONS file attribute specifies the number of read heads on a reader-sorter device.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 3 inclusive, and specifies the number of read heads on the reader-sorter device.

Examples:

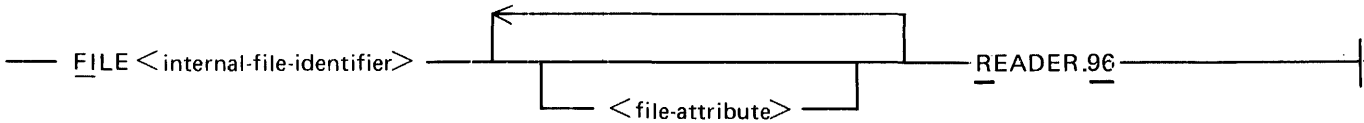
EXECUTE A/B; FILE SORTER READER.SORTER.STATIONS = 2;

MODIFY PROG1; FILE CHECKS RST 1;

READER.96

The `READER.96` file attribute specifies that input to the file is from a 96-column card reader.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER READER.96;
```

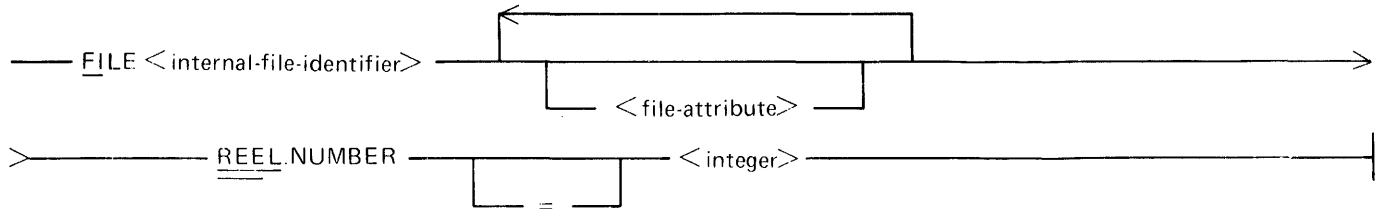
```
MODIFY PROG1; FILE INPUT R96;
```


FILE REEL.NUMBER

REEL.NUMBER

The REEL.NUMBER file attribute specifies the number of the first reel of a multireel tape file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field determines the first reel number for a multireel tape file.

Examples:

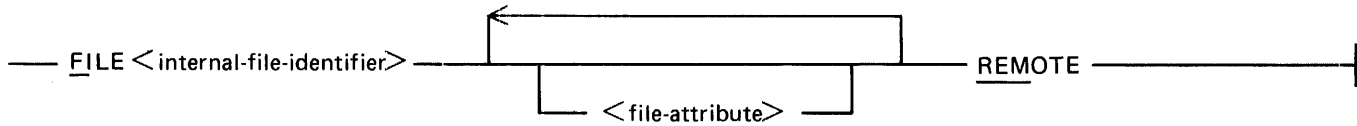
EXECUTE A/B; FILE TAPEFILE REEL.NUMBER = 1;

MODIFY PROG1; FILE PAYROLL REEL 3;

REMOTE

The REMOTE file attribute ^{declares} ~~declares~~ a file as a remote file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

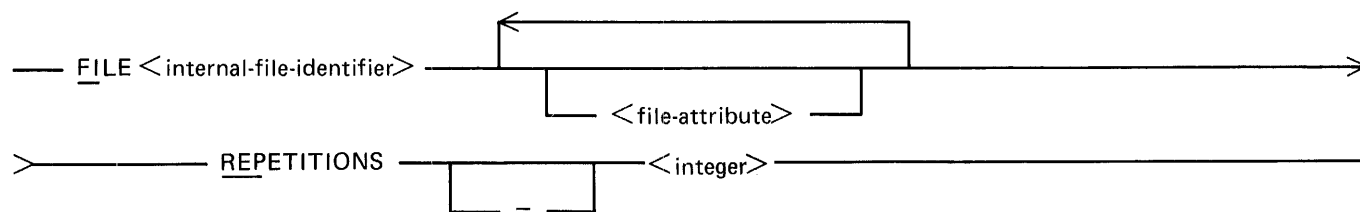
EXECUTE A/B; FILE MASTER REMOTE;

MODIFY PROG1; FILE INPUT REM;

REPETITIONS

The REPETITIONS file attribute specifies the number of copies of a backup file to be printed or punched by the SYSTEM/BACKUP program. The value of the REPETITIONS file attribute can be overridden by the COPIES option of the PB system command (described in section 5) and is valid only at file creation time. If the REPETITIONS value for a printer or punch file is greater than 1, the file is automatically directed to backup by the MCP, unless explicitly prevented by specifying the NO BACKUP file attribute.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 1 to 63 inclusive, and specifies the number of copies of the file to be printed or punched.

Examples:

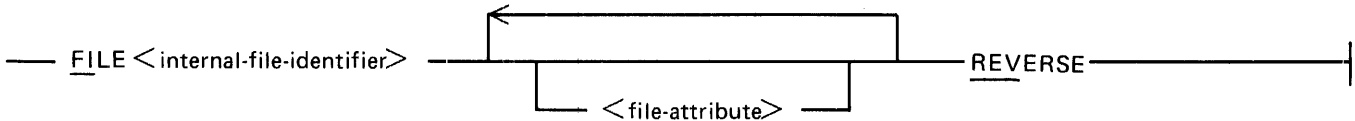
EXECUTE A/B; FILE PRINTER REPETITIONS = 3;

MODIFY PROG1; FILE CHECKS REP 2;

REVERSE

The REVERSE file attribute applies to tape files only and causes the tape file to be read or written in reverse. The REVERSE file attribute sets the REVERSE attribute for an implied open of a file. An implied open of a file occurs when a read or write operation is performed on a file that was not explicitly opened. Implied open attributes are ignored if the file is explicitly opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

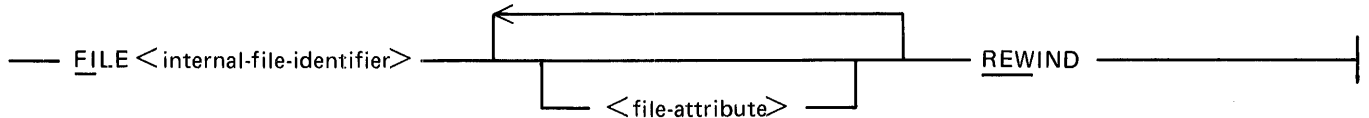
```
EXECUTE A/B; FILE TAPEFILE REVERSE;
```

```
MODIFY PROG1; FILE TAPE REV;
```

REWIND

The REWIND file attribute causes the tape file to be rewound when the file is opened and applies to a tape file that is implicitly opened. An implied open of a file occurs when a read or write operation is performed on a file that was not explicitly opened. If the file is not explicitly opened, then the implied open attributes are ignored.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

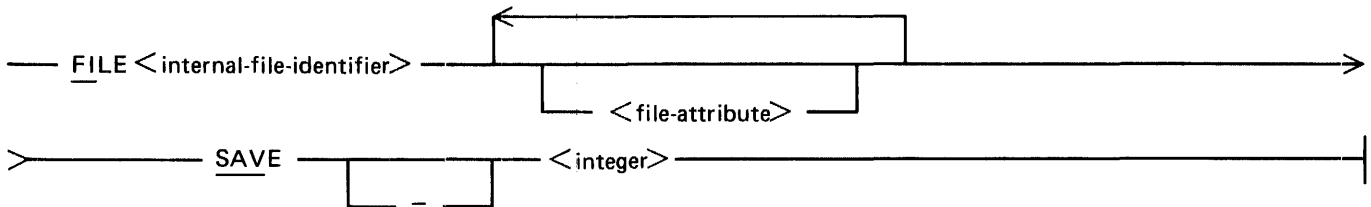
```
EXECUTE A/B; FILE TAPEFILE REWIND;
```

```
MODIFY PROG1; FILE TAPE NO REW;
```

SAVE

The SAVE file attribute specifies the number of days a tape or disk file can be saved. This file attribute is of visual value only.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field specifies the number of days to be added to the file creation date to determine the last day for the file to be saved.

Examples:

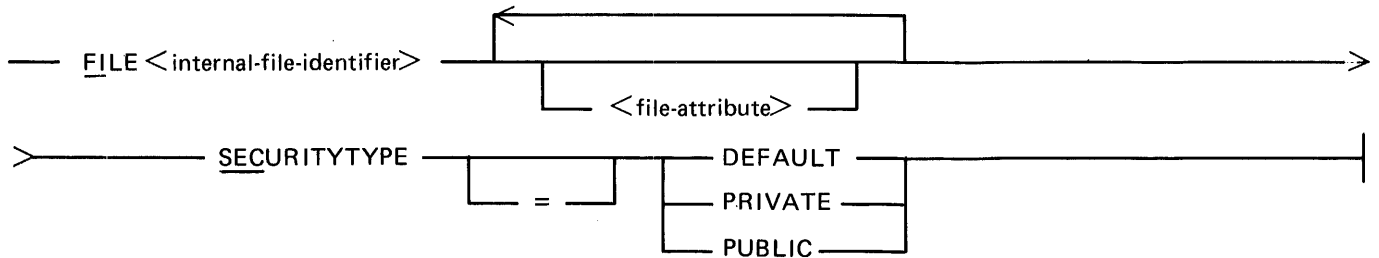
```
EXECUTE A/B; FILE MASTER SAVE = 30;
```

```
MODFIY PROG1; FILE PAYROLL SAV 10;
```

SECURITYTYPE

The SECURITYTYPE file attribute designates the type of security for a disk file. A new disk file is given the security type specified by the File Parameter Block (FPB) when it is closed and entered in the disk directory.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

DEFAULT

The DEFAULT keyword causes the file to use the default security type.

PRIVATE

The PRIVATE keyword causes the file to be a private file. Only processes with a privileged user-code can access this file.

PUBLIC

The PUBLIC keyword causes the file to be a public file and can be accessed by any process.

Examples:

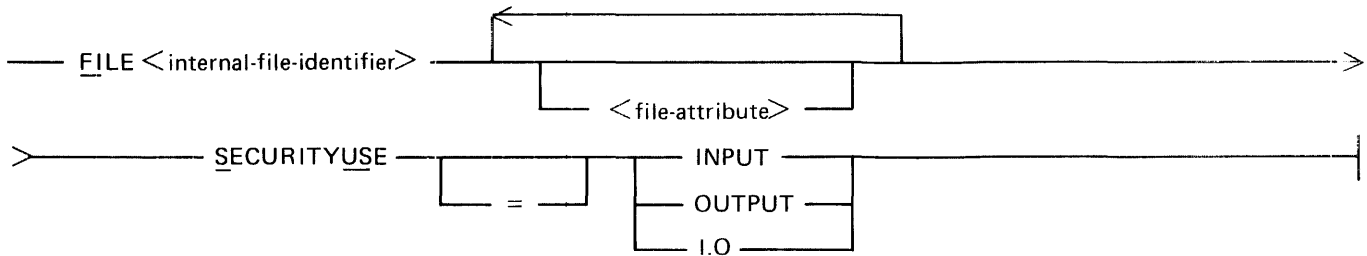
```
EXECUTE A/B; FILE MASTER SECURITYTYPE = PRIVATE;
```

```
MODIFY PROG1; FILE PAYROLL SEC DEFAULT;
```

SECURITYUSE

The SECURITYUSE file attribute designates the type of input/output (I/O) allowed for the file. A new disk file is given the security I/O code specified by the File Parameter Block (FPB) when it is closed and entered in the disk directory.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

INPUT

The INPUT keyword causes the security I/O to be input only.

OUTPUT

The OUTPUT keyword causes the security I/O to be output only.

I.O

The I.O keysymbol causes the security I/O to be input and output.

Examples:

```
EXECUTE A/B; FILE MASTER SECURITYUSE = INPUT;
```

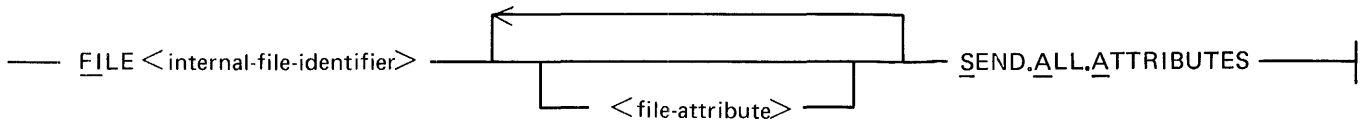
```
MODIFY PROG1; FILE PAYROLL SUS OUTPUT;
```


FILE SEND.ALL.ATTRIBUTES

SEND.ALL.ATTRIBUTES

The SEND.ALL.ATTRIBUTES file attribute specifies to the BNA logical input/output (I/O) process, the BNA/HSLIO program, that all of the file attributes of this file are to be sent to the cooperating BNA host system.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

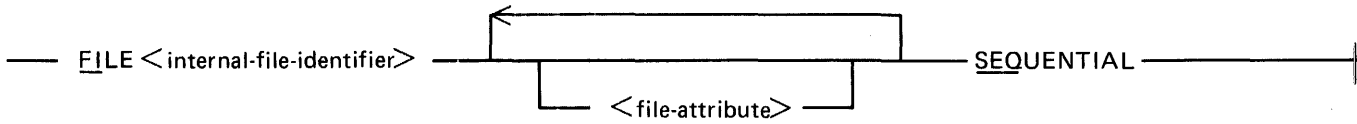
EXECUTE A/B; FILE MASTER SEND.ALL.ATTRIBUTES;

MODIFY PROG1; FILE PAYROLL SAA;

SEQUENTIAL

The SEQUENTIAL file attribute specifies that the file access mode is sequential.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER SEQUENTIAL;
```

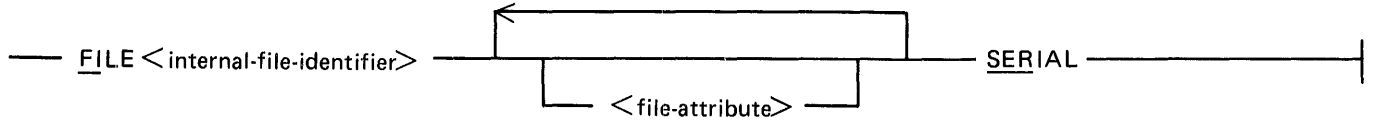
```
MODIFY PROG1; FILE INPUT SEQ;
```

FILE SERIAL

SERIAL

The SERIAL file attribute specifies that the file is to be processed serially.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

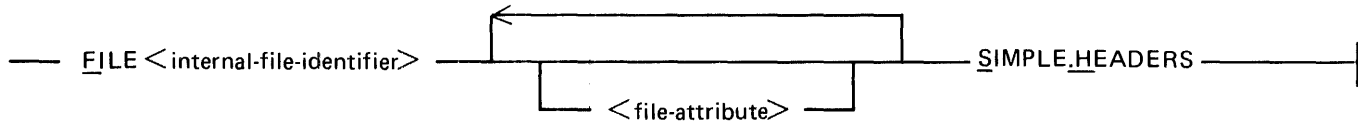
EXECUTE A/B; FILE MASTER SERIAL;

MODIFY PROG1; FILE PAYROLL SER;

SIMPLE.HEADERS

The SIMPLE.HEADERS file attribute causes the remote file to use the 50-byte NDL message header as an extended remote key, but to have no Message Control System (MCS) control functions. Refer to the B 1000 Systems Network Definition Language (NDL) Language Manual for a complete description of the functions of a remote file with simple headers.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

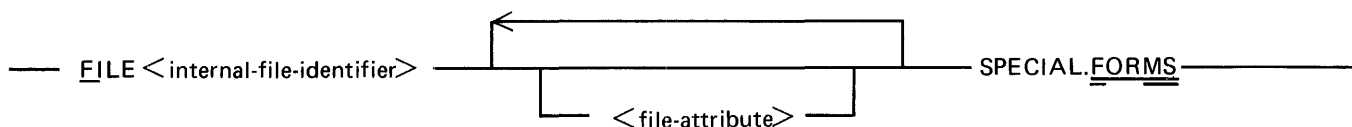
```
EXECUTE A/B; FILE MCSQUEUE SIMPLE.HEADERS;
```

```
MODIFY PROG1; FILE RMTEFILE S.H;
```

SPECIAL.FORMS

The SPECIAL.FORMS file attribute causes the program to be suspended and causes the MCP to display a message requesting the operator to load special forms in a device (line printer or card punch) before the file is opened.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

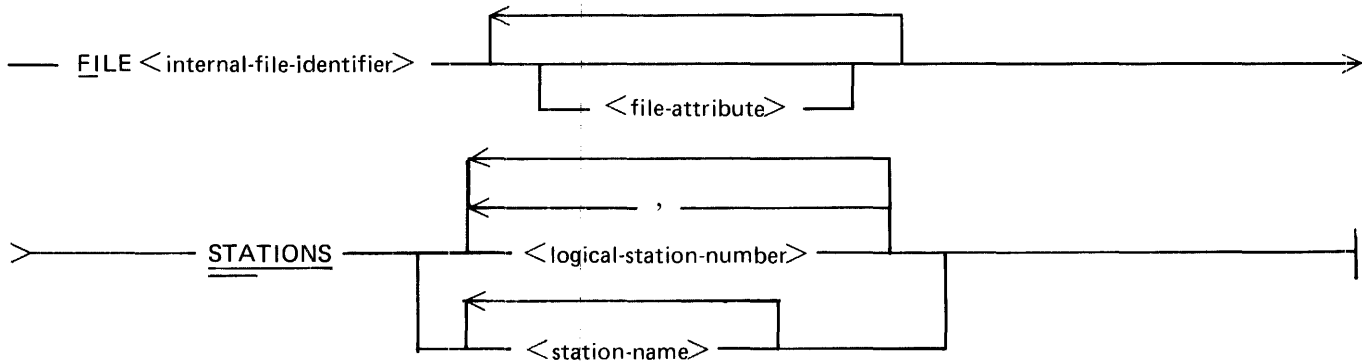
```
EXECUTE A/B; FILE LINE SPECIAL.FORMS;
```

```
MODIFY PROG1; FILE CHECKS FORMS;
```

STATIONS

The STATIONS file attribute applies to files with a device type of REMOTE and specifies the stations to be attached to this remote file at program-execution time. The STATIONS file attribute is not valid for the MODIFY program control instruction.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

logical-station-number

This field can contain any logical station number (LSN) that is declared in the network controller and specifies the LSN that is attached to the remote file of the program at execution time. A maximum of 999 logical station numbers can be specified with the STATIONS file attribute.

station-name

This field can contain any valid station name that is declared in the network controller and specifies the station that is attached to the remote file of the program at execution time. A maximum of 299 station names can be specified with the STATIONS file attribute.

Examples:

```
EXECUTE A/B; FILE RMTE STATIONS MT01, ET02, TD03;
```

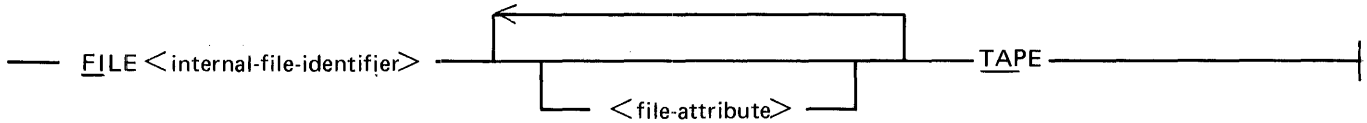
```
EXECUTE PROG1; FILE MCSQUEUE STA 1 5 7 9;
```

FILE TAPE

TAPE

The TAPE file attribute specifies that the output file is to be written to a tape device. The TAPE file attribute is the most general of the tape device file attributes.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

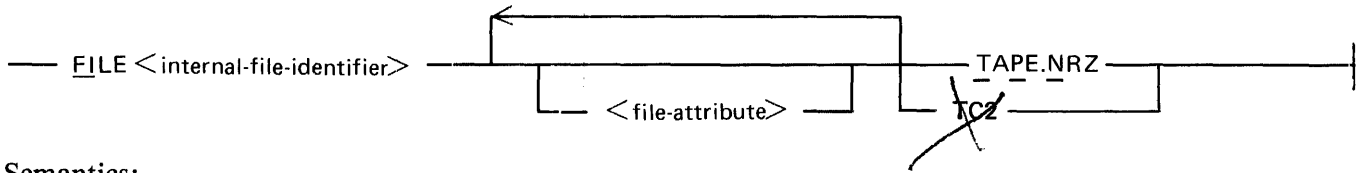
```
EXECUTE A/B; FILE TAPEFILE TAPE;
```

```
MODIFY PROG1; FILE PAYROLL TAP;
```

TAPE.NRZ

The TAPE.NRZ file attribute specifies that the output file is to be written to a tape device using Non-Return-to-Zero (NRZ) tape mode only.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

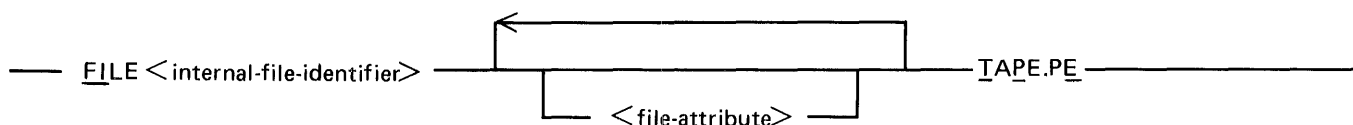
EXECUTE A/B; FILE TAPEFILE TAPE.NRZ;

MODIFY PROG1; FILE PAYROLL TPN;

TAPE.PE

The TAPE.PE file attribute specifies that the output file is written to a tape device using Phase-Encoded (PE) tape mode only. Records for TAPE.PE files must be of an even size because PE tapes use even size blocks.

Syntax:



Semantics:

internal-file-identifier

This field contains any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field contains any valid file attribute described in this section.

Examples:

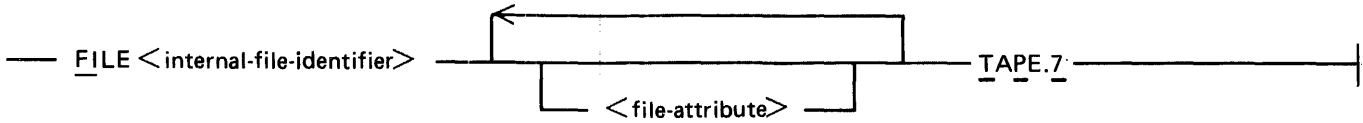
```
EXECUTE A/B; FILE TAPEFILE TAPE.PE;
```

```
MODIFY PROG1; FILE PAYROLL TPE;
```

TAPE.7

The TAPE.7 file attribute specifies that the output file is to be written to a tape device using 7-track tape mode only.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE TAPEFILE TAPE.7;
```

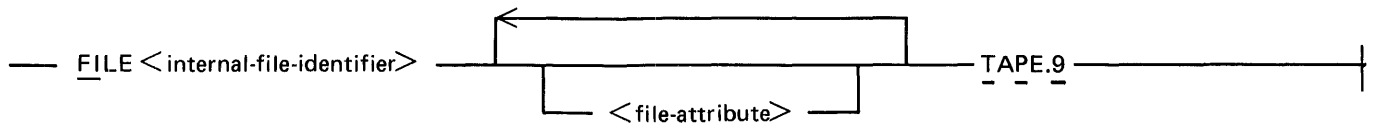
```
MODIFY PROG1; FILE PAYROLL TP7;
```

FILE TAPE.9

TAPE.9

The TAPE.9 file attribute specifies that the output file is to be written to a tape device using 9-track tape mode only.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

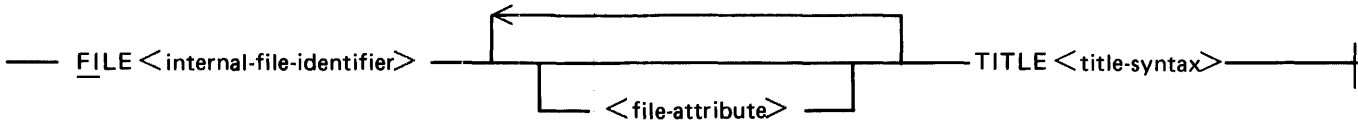
```
EXECUTE A/B; FILE TAPEFILE TAPE.9;
```

```
MODIFY PROG1; FILE PAYROLL TP9;
```

TITLE

The TITLE file attribute specifies the external name of a file using syntax of the form B/C ON A, where B is the name of the first part of the file name, C is the name of the second part of the file name, and A is the family name (or disk identifier).

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

title-syntax

This field refers to the external name of the file, using the form B/C on A.

Examples:

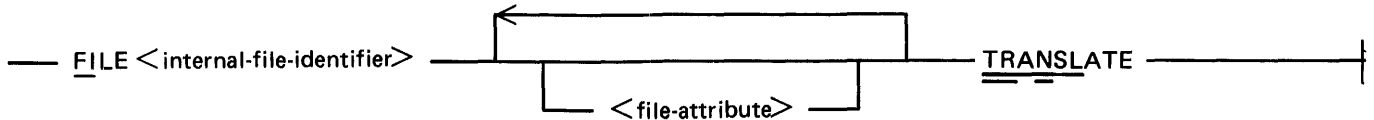
EXECUTE A/B; FILE MASTER TITLE MASTER/FILE ON PACKA;

MODIFY PROG1; FILE PAYROLL TITLE PROG1 ON TEST;

TRANSLATE

The TRANSLATE file attribute specifies that the file is to be translated using the MCP soft translate facility. Translation is not allowed on input/output files. The name of the translate file to be used must be specified by the TRANSLATE.FILE.NAME file attribute.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

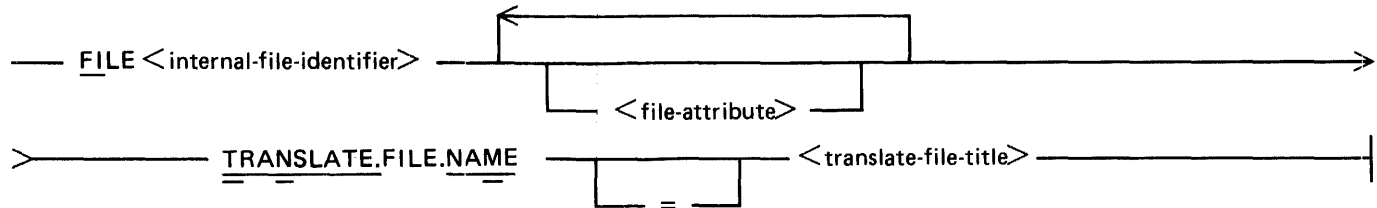
EXECUTE A/B; FILE PRINTER TRANSLATE;

MODIFY PROG1; FILE LINE TRN;

TRANSLATE.FILE.NAME

The TRANSLATE.FILE.NAME file attribute specifies the file name of the file to be used, if soft translate is requested. The first name portion of the file name is assumed to be TRANSLATE and is not specified with this file attribute. The translate file must have been created using the CREATE/TABLE program and must reside on the SYSTEM disk.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

translate-file-title

This field can contain any valid B 1000 file title and specifies the name of the translate file to be used.

Examples:

```
EXECUTE A/B; FILE PRINTER TRANSLATE.FILE.NAME = TRANSFILE;
```

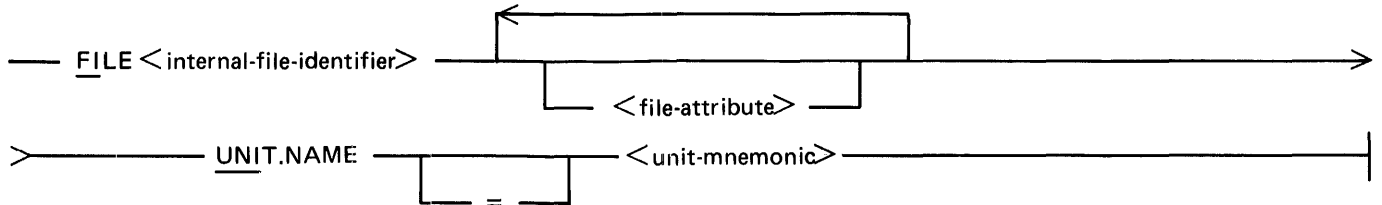
```
MODIFY PROG1; FILE PAYROLL TNM PAYTRAN;
```

FILE UNIT.NAME

UNIT.NAME

The UNIT.NAME file attribute directs the file to the specified unit mnemonic. If a family name is specified for the file, then this file attribute is ignored. This file attribute does not apply to disk files.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

unit-mnemonic

This field can contain any valid B 1000 unit-mnemonic. Refer to section 1 for a list of the valid unit mnemonics for the B 1000 system.

Examples:

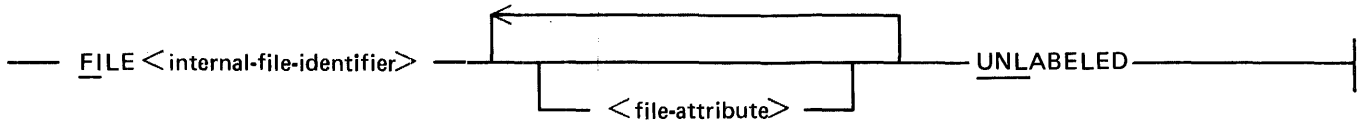
EXECUTE A/B; FILE OUTPUT UNIT.NAME MTA;

MODIFY PROG1; FILE LINE UNI LPB;

UNLABELED

The UNLABELED file attribute specifies that there is no label on this file.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE MASTER UNLABELED;
```

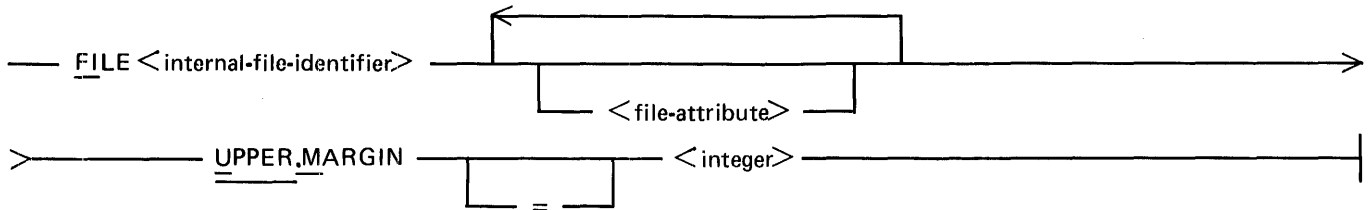
```
MODIFY PROG1; FILE INPUT UNL;
```


FILE UPPER.MARGIN

UPPER.MARGIN

The UPPER.MARGIN file attribute specifies the number of lines from the top of the form to the first print line.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

integer

This field can contain any integer in the range 0 to 255 inclusive, and specifies the number of lines from the top of the form to the first print line for this line printer file.

Examples:

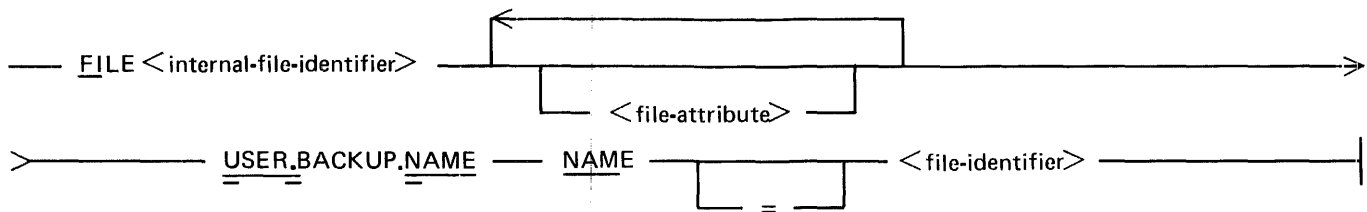
EXECUTE A/B; FILE PRINT UPPER.MARGIN = 4;

MODIFY PROG1; FILE CHECKS U.M 2;

USER.BACKUP.NAME

The USER.BACKUP.NAME file attribute applies to disk, printer, and punch backup files and specifies the external file identifier to be used rather than the default backup file identifier of BACKUP/PRT<integer> or BACKUP/PCH<integer> when the file is entered in the disk directory. This file attribute is ignored for backup files sent to tape. The NAME file attribute must be specified to assign the alternate name for the backup file identifier.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

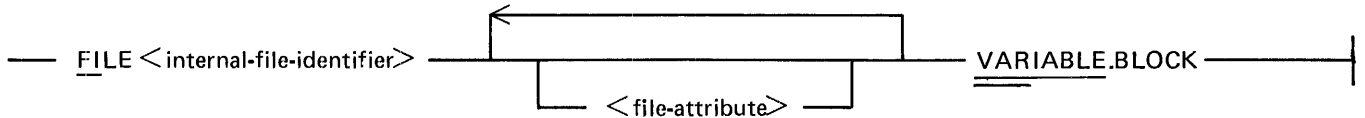
```
EXECUTE A/B; FILE LINE USER.BACKUP.NAME NAME A/#LINE
```

```
MODIFY PROG1; FILE CHECKS U.N NAM CHECKS;
```

VARIABLE.BLOCK

The VARIABLE.BLOCK file attribute causes the file to be processed using variable-length records.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the `<file-attribute>` is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

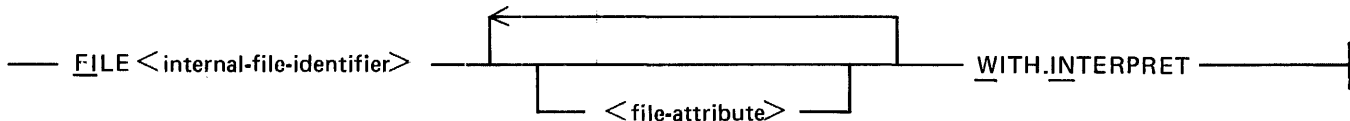
EXECUTE A/B; FILE MASTER VARIABLE.BLOCK;

MODIFY PROG1; FILE PAYROLL VAR;

WITH.INTERPRET

The WITH.INTERPRET file attribute applies to files that are opened implicitly, and sets the data recorder INTERPRET attribute. An implied open of a file occurs when a program performs a read or write operation on a file without explicitly opening the file. Implied open attributes are ignored if the file is opened explicitly.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

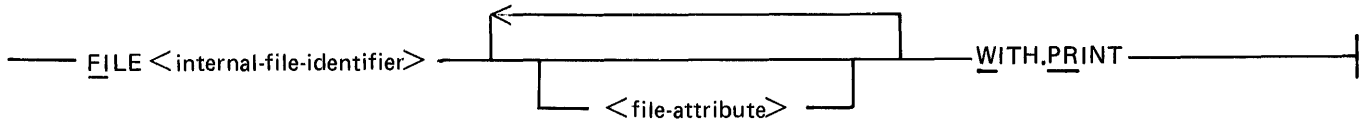
```
EXECUTE A/B; FILE PUNCHFILE WITH.INTERPRET;
```

```
MODIFY PROG1; FILE PAYPUNCH WIN;
```

WITH.PRINT

The WITH.PRINT file attribute applies to files that are opened implicitly, and sets the data recorder PRINT attribute. An implied open of a file occurs when a program performs a read or write operation on a file without explicitly opening the file. Implied open attributes are ignored if the file is opened explicitly.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

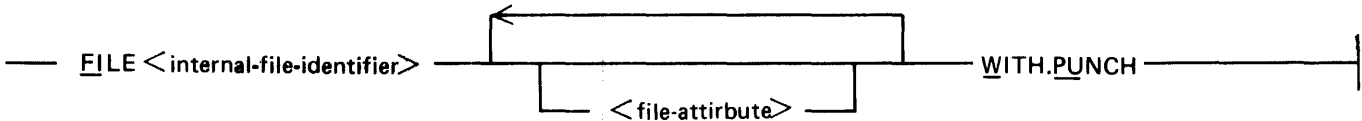
```
EXECUTE A/B; FILE PUNCHFILE WITH.PRINT;
```

```
MODIFY PROG1; FILE PAYPUNCH WPR;
```

WITH.PUNCH

The WITH.PUNCH file attribute applies to files that are opened implicitly, and sets the data recorder PUNCH attribute. An implied open of a file occurs when a program performs a read or write operation on a file without explicitly opening the file. Implied open attributes are ignored if the file is opened explicitly.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

```
EXECUTE A/B; FILE PUNCHFILE WITH.PUNCH;
```

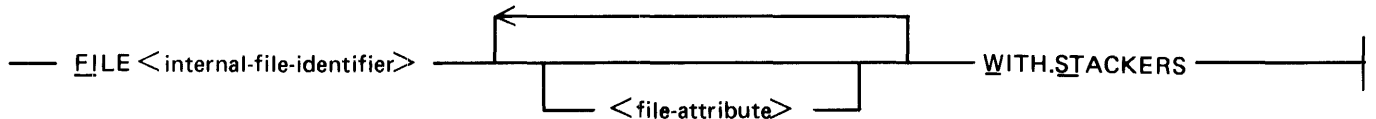
```
MODIFY PROG1; FILE PAYPUNCH WPU;
```

FILE WITH.STACKERS

WITH.STACKERS

The WITH.STACKERS file attribute applies to files that are opened implicitly, and sets the data recorder STACKERS attribute. An implied open of a file occurs when a program performs a read or write operation on a file without explicitly opening the file. Implied open attributes are ignored if the file is opened explicitly.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the <file-attribute> is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

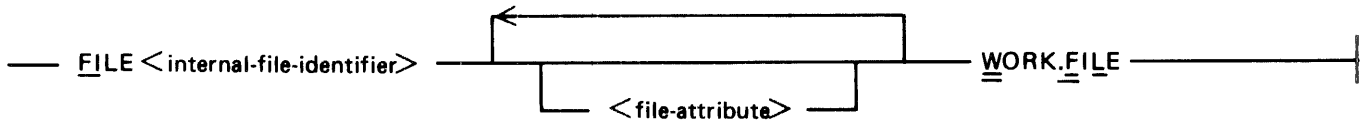
EXECUTE A/B; FILE PUNCHFILE WITH.STACKERS;

MODIFY PROG1; FILE PAYPUNCH WST;

WORK.FILE

The **WORK.FILE** file attribute assigns the file as a work file to be used internally. To make the file name unique, a work file has the mix number of the program substituted for part of the first name.

Syntax:



Semantics:

internal-file-identifier

This field can contain any valid internal file identifier in a program and specifies the file for which the **<file-attribute>** is being changed.

file-attribute

This field can contain any valid file attribute described in this section.

Examples:

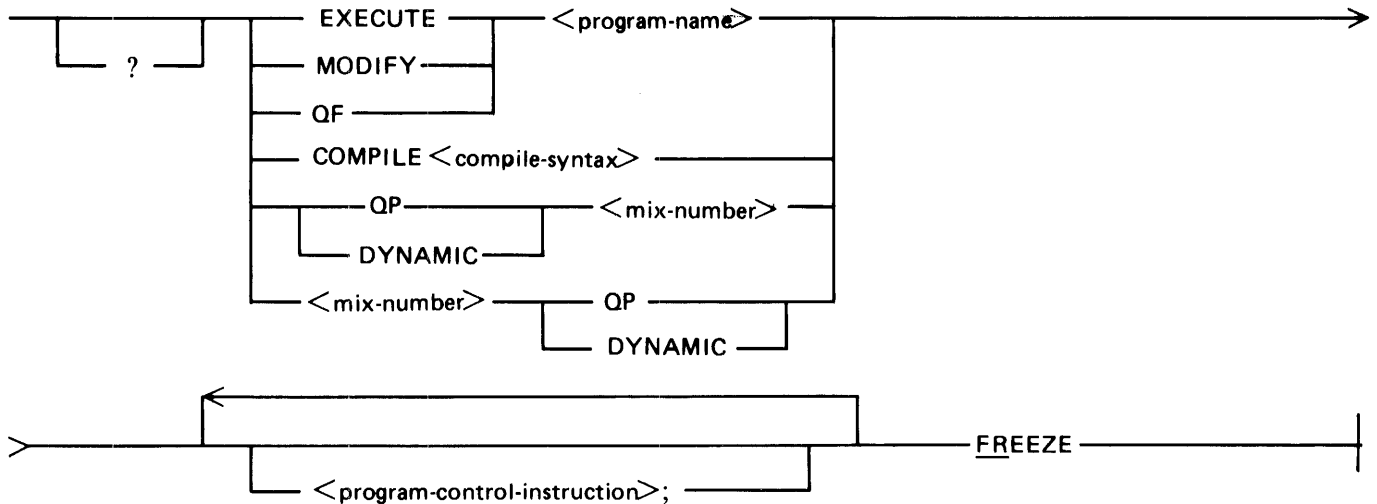
```
EXECUTE A/B; FILE WORK1 WORK.FILE;
```

```
MODIFY PROG1; FILE PAYWORK WFL;
```


FREEZE

The FREEZE program control instruction prohibits the rolling out to disk of the base-to-limit area of a program during its execution. The base-to-limit area remains in the same memory location until the program goes to end of job (EOJ). The base-to-limit area of a program contains the data for the program. The FREEZE status can be removed using the UNFREEZE program control instruction.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

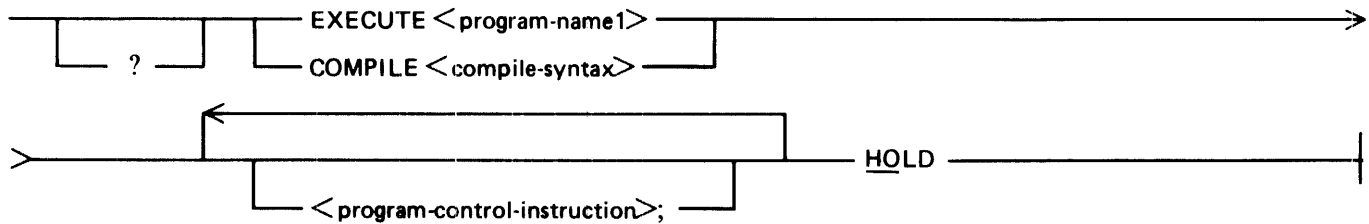
```
EXECUTE A/B FREEZE;
```

```
1236 DY FREEZE
```

HOLD

The HOLD program control instruction places a program in the waiting schedule, thus prohibiting its execution until it is forced into the active schedule by the FS system command described in section 5.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

Examples:

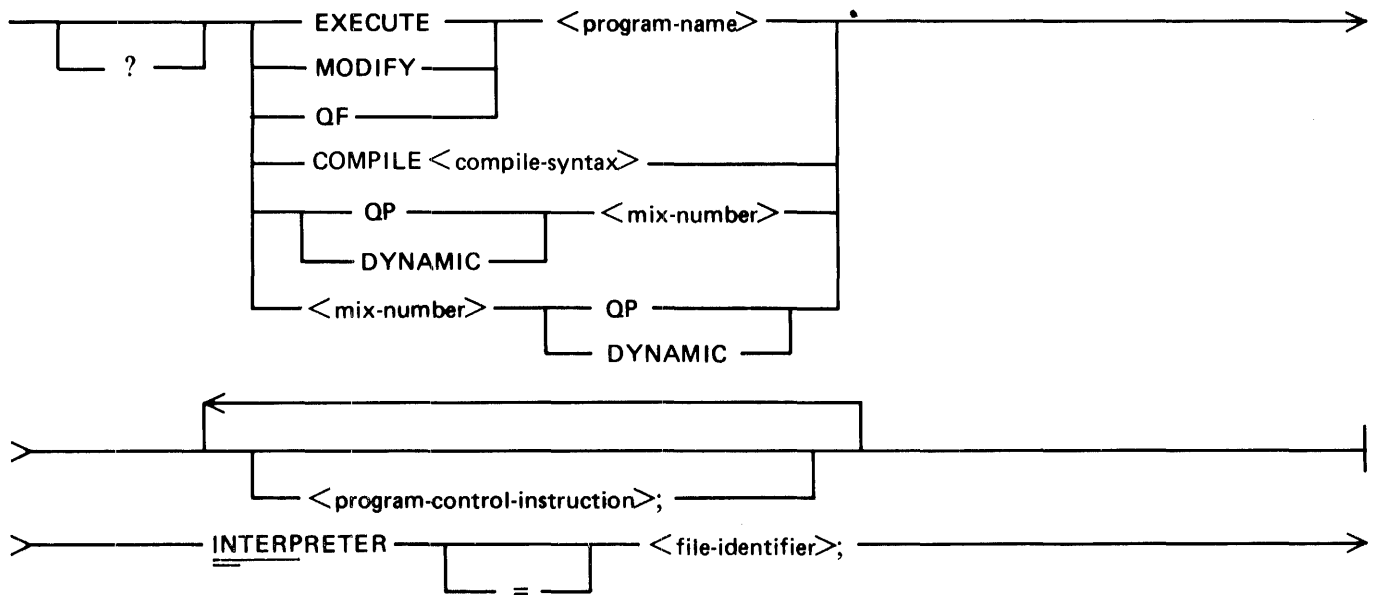
```
EXECUTE DMPALL HOLD;
```

```
COMPILE MILL/GEARS FORTRAN LI HO;
```

INTERPRETER

The INTERPRETER program control instruction specifies the interpreter to be used by the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

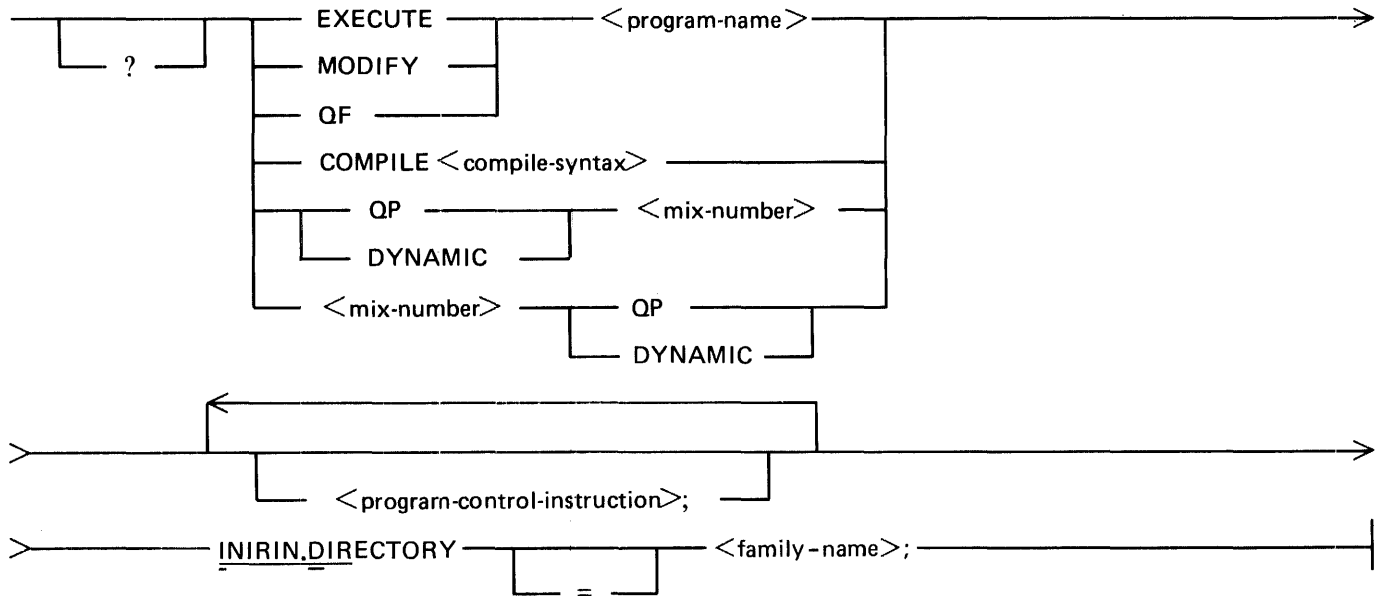
```
EXECUTE BETA/ALPHA INTERPRETER COBOL/INTERP001;
```

```
EX X/Y IN CCC/SDL/INTERP3;
```

INTRIN.DIRECTORY

The INTRIN.DIRECTORY program control instruction causes the MCP to use a specified user pack to locate the intrinsic files.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

family-name

This field can contain any valid 10-character B 1000 disk identifier and specifies the name of the user disk on which the intrinsic files are located.

Examples:

```
EXECUTE ALPH/BETA; INTRIN.DIRECTORY = UTILPACKA;
```

```
EX B; ID USER;
```

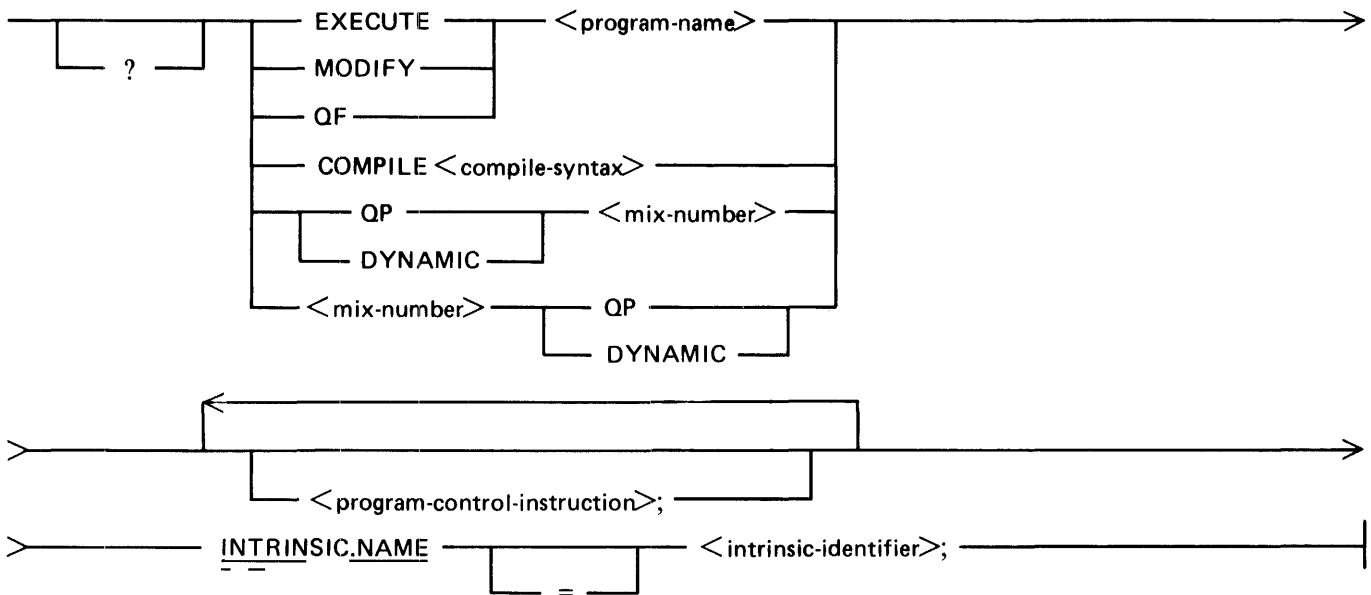
INTRINSIC.NAME

INTRINSIC.NAME

The INTRINSIC.NAME program control instruction specifies the first name portion of the file identifier for an intrinsic file that is requested by the program.

The subfile directory portion of the file identifier for an intrinsic file named AGGREGATE cannot be changed.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

intrinsic-identifier

This field can contain a 10-character string and specifies the first name portion of the file identifier for the intrinsic file.

Examples:

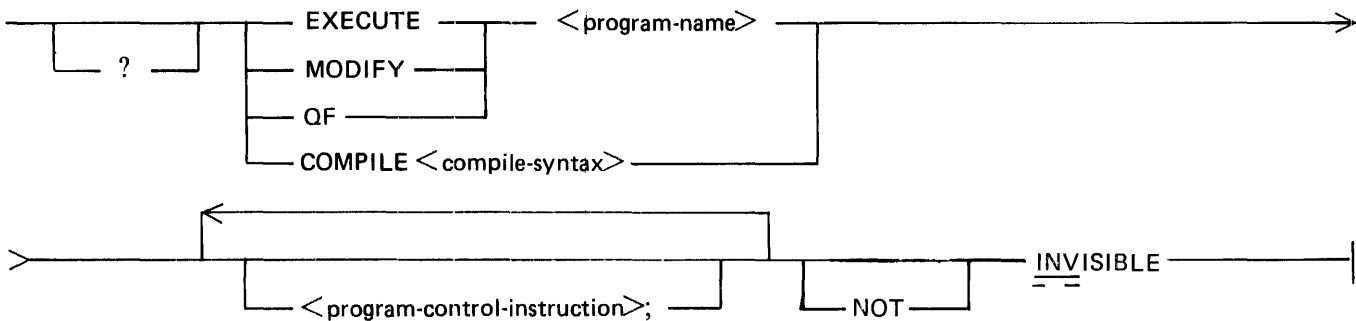
```
EXECUTE ALPHA/BETA; INTRINSIC.NAME ZZZ.INTRIN;
```

```
EX B; IT TEST.INTRIN;
```

INVISIBLE

The INVISIBLE program control instruction causes the task and its associated information to not appear in the output of the MX or WY system commands described in section 5. This instruction is intended to minimize ODT traffic by hiding system-oriented programs that remain in the mix but whose status is not usually of interest to the operator. The INVISIBLE flag can be set and reset using the IV system command described in section 5.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, or COMPILE program control instruction.

Examples:

```
EXECUTE A/B; INVISIBLE
```

```
MODIFY TEST INV
```

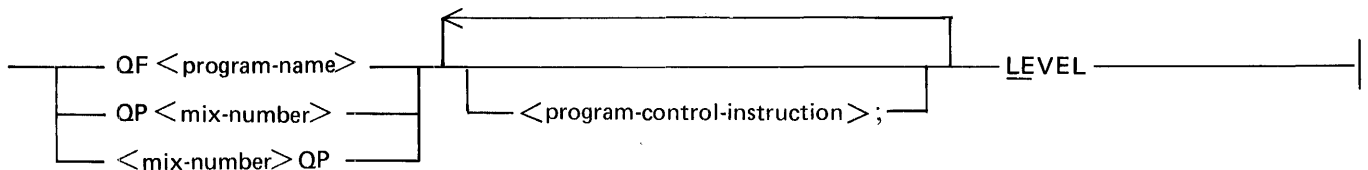
LEVEL

The LEVEL program control instruction queries the compiler level of a program.

The compiler level indicates which S-operators generated by the compiler are valid for the associated interpreter. If an S-operator is changed or removed from the interpreter, the level of the compiler is increased by one and the user is required to recompile the programs using that interpreter. The compiler level is not affected by the addition of a new S-operator. New S-operators are usually added as new features in the compiler attributes.

When a program goes to beginning of job, the MCP ensures that the program and its associated interpreter have the same level. This check can be bypassed using the OVERRIDE program control instruction.

Syntax:



Semantics:

program-name

This field can contain any program name that is currently in the disk directory.

mix-number

This field can contain any mix number of a program currently executing.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the QF or QP program control instruction.

Examples:

QF DMPALL LEVEL;

123 QP LE;

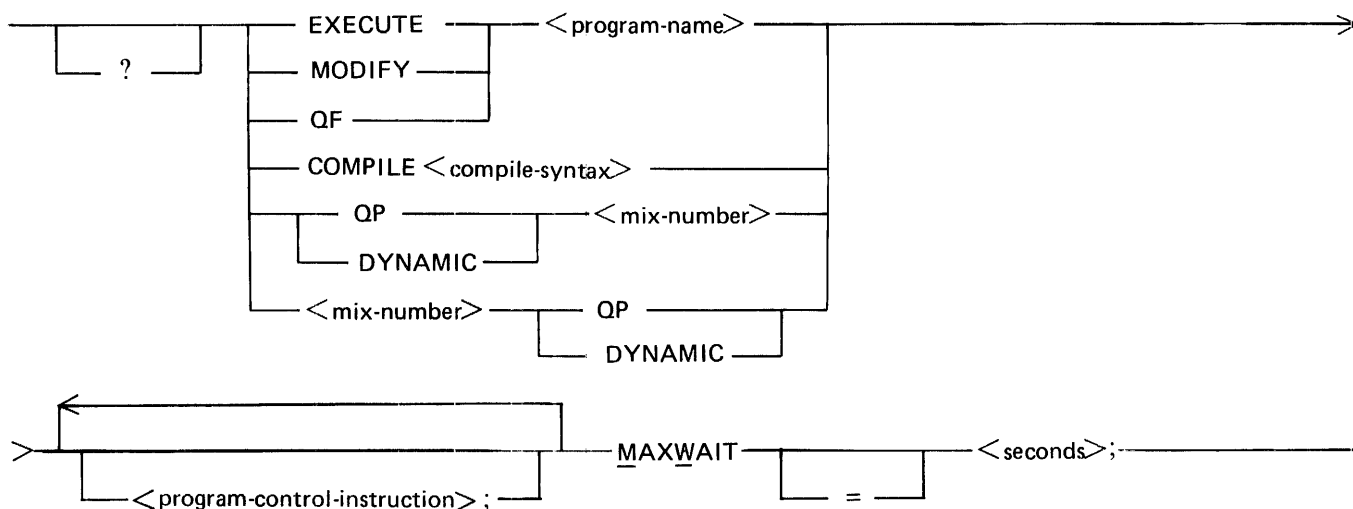
QP 5628 LEVEL;

MAXWAIT

The MAXWAIT program control instruction specifies the maximum amount of time, in seconds, that a DMS program waits on contention for a locked record.

If the value specified by the MAXWAIT program control instruction is exceeded, a deadlock exception condition is returned to the DMS program. The default value is zero. The default setting causes the DMS program to use the MAXWAIT value associated with the data base that it opens. The default MAXWAIT value for a data base is 180 seconds.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

seconds

This field can contain any integer and specifies the number of seconds to wait on contention for a locked record.

Examples:

```
EXECUTE WORK/DMS/BATCH; MAXWAIT = 20;
```

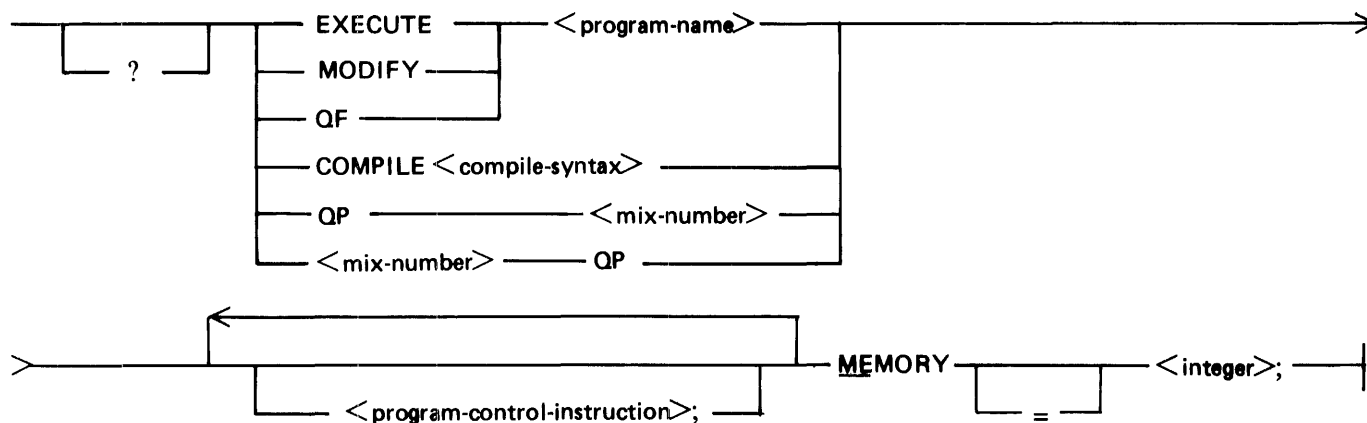
```
EX TEST; MW 5;
```


MEMORY

The MEMORY program control instruction overrides the dynamic memory size assigned by the compiler for a program at execution time.

The program remains in the active schedule if there is not enough dynamic memory available to run the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, or QP program control instruction.

integer

This field can contain any integer in the range 0 to 16,777,208 inclusive, and specifies the amount of dynamic memory in bits for use by the program.

Examples:

```
COMPILE DIVIDENDS COBOL SYNTAX; MEMORY = 50000;
```

```
EXECUTE DOCUMENT/EDITOR; ME 40000;
```

MEMORY.PRIORITY

The MEMORY.PRIORITY program control instruction assigns a priority to code segments in memory for a program.

The MEMORY.PRIORITY program control instruction is valid only when the Priority Memory Management algorithm is being used by the MCP. This algorithm is set with the MPRI MCP option.

When a program code segment is read into memory by the MCP, the memory space it occupies is given an initial priority equal to the value specified by the MEMORY.PRIORITY program control instruction. Program code segments of one program cannot overlay those of another program that have a higher memory priority, thus allowing more important program code segments to be protected. However, program code segments that are not referenced by the program for a period of time decay to a lower memory priority. If the memory priority becomes low enough, the program code segment is overlaid. Usually, the period of time that causes the program code segment to decay is equal to 1.5 times the SAMPLING.INTERVAL value, unless the SECONDS.BEFORE.DECAY program control instruction specifies a different interval for program code segments that are marked important. If a program code segment is accessed by a program at any time before being overlaid, its memory priority is restored to the original value.

A MEMORY.PRIORITY value of 9 or greater is referred to as a crashout priority, and has many additional effects. If insufficient overlayable memory space for a request having a crashout priority is available, the MCP deallocates save memory space having a lower memory priority. Such a deallocation is performed on the run structure (base-to-limit area) of a lower-priority program, and results in an abbreviated rollout of the program selected as the victim. This action performed by the MCP is called crashout, and it suspends the victim and writes only the base-to-limit area of the program (not any file or code memory space) to temporary disk storage. It then makes the space occupied by the run structure available to satisfy the memory request. The MCP periodically (at each N.SECOND interval) reinstates any victim programs that were crashed out.

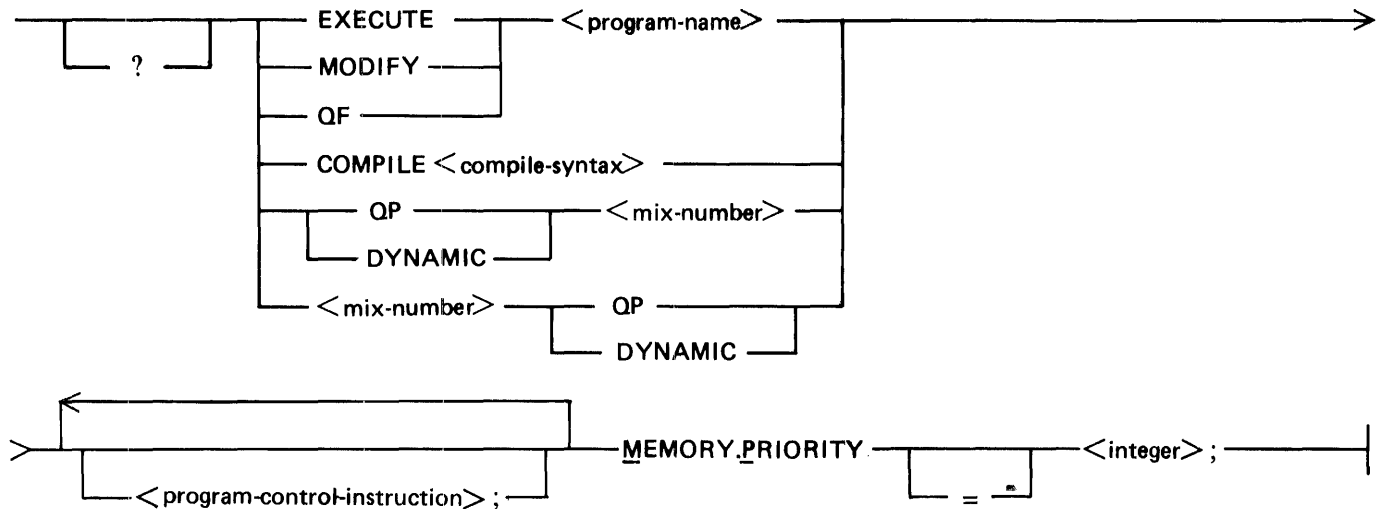
Entering a program having a crashout priority in the active schedule does not cause any crashout actions to be taken on running programs in order to begin the high-priority task. Crashout can be caused only by an executing program having a MEMORY.PRIORITY value of nine or greater, and whose MEMORY.PRIORITY value is higher than that of the program that is to be crashed out. For example, a program with a MEMORY.PRIORITY value of 12 cannot cause crashout on any other program with a MEMORY.PRIORITY value of 12 or above, but can cause any program with a MEMORY.PRIORITY of less than 12 to be crashed out.

After a program has gone to beginning of job, any queries or changes to the MEMORY.PRIORITY value in the working copy of the program must be accomplished through the MP system command described in section 5.

Refer to appendix A for a complete description of the MCP Memory Management mechanism.

MEMORY.PRIORITY

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 15 inclusive, and specifies the memory priority assigned to program code segments.

Examples:

EXECUTE A/B; MEMORY.PRIORITY = 8;

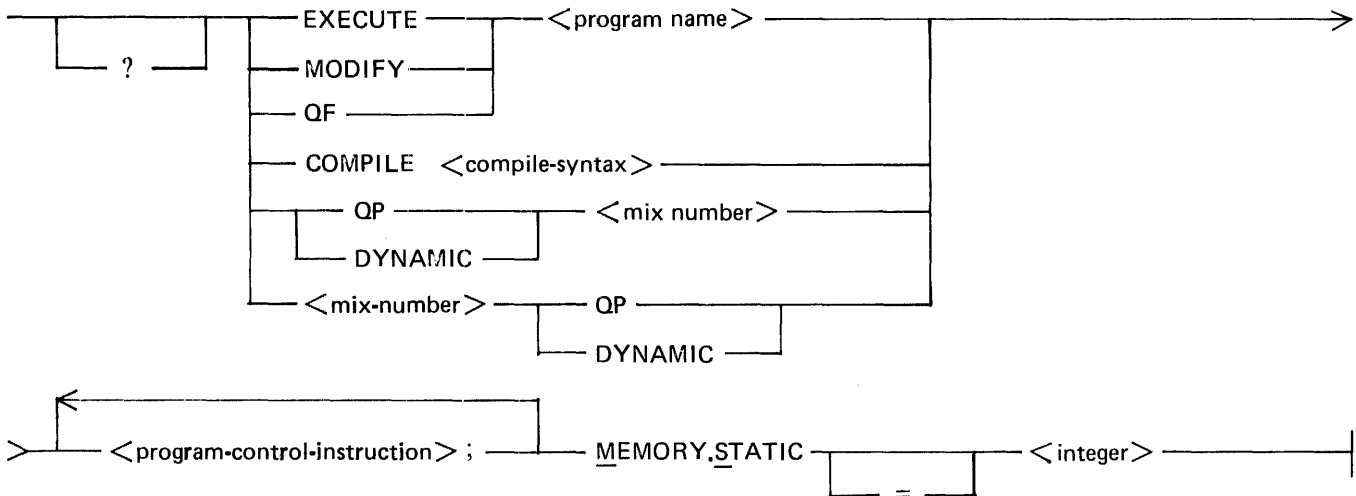
EX NETWORK/CONTROLLER; MP 15;

MEMORY.STATIC

The MEMORY.STATIC program control instruction overrides the default static memory size of a program at execution time.

When MEMORY.STATIC is used with COMPILE statements, the static memory is reserved for the compiler, not the program compiled.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled to be executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 16,777,208 inclusive, and specifies the amount of static memory in bits for use by the program.

Examples:

```
COMPILE DIVIDENDS COBOL SYNTAX;MS = 50000
```

```
EX SCHEDULES;MS = 330000;
```

MODIFY

The MODIFY program control instruction permanently changes the attributes of an object program using other program control instructions.

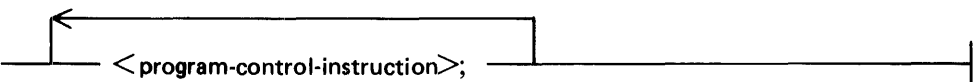
The MODIFY program control instruction must be the first of a series of program control instructions pertaining to the modification of the object program.

If the object program referenced in the MODIFY program control instruction resides on a user disk, the disk identifier must be part of <program-name> in order for the MCP to locate the correct file.

The values specified by the following program control instructions can be changed using the MODIFY program control instruction.

CHARGE	PRIORITY
CONDITIONAL	PROCESSOR.PRIORITY
FILE	PROTECTED
FREEZE	RR
INTERPRETER	SCHEDULE.PRIORITY
INTRIN.DIRECTORY	SECONDS.BEFORE.DECAY
INTRINSIC.NAME	SWITCH
INVISIBLE	TIME
MAXWAITDIRECTORY	UNCONDITIONAL
INTRINSIC.NAME	UNFREEZE
MEMORY.PRIORITY	UNOVERRIDE
NODIFIT	VIRTUAL.DISK
OVERRIDE	

Syntax:

— MODIFY <program-name> —  <program-control-instruction>; —

Semantics:

program-name

This field can contain any valid B 1000 object program name that follows the B 1000 file-naming conventions and specifies the program to be modified.

program-control-instruction

This field can contain any valid program control instruction described in this section that is allowed to follow a MODIFY program control instruction.

Example:

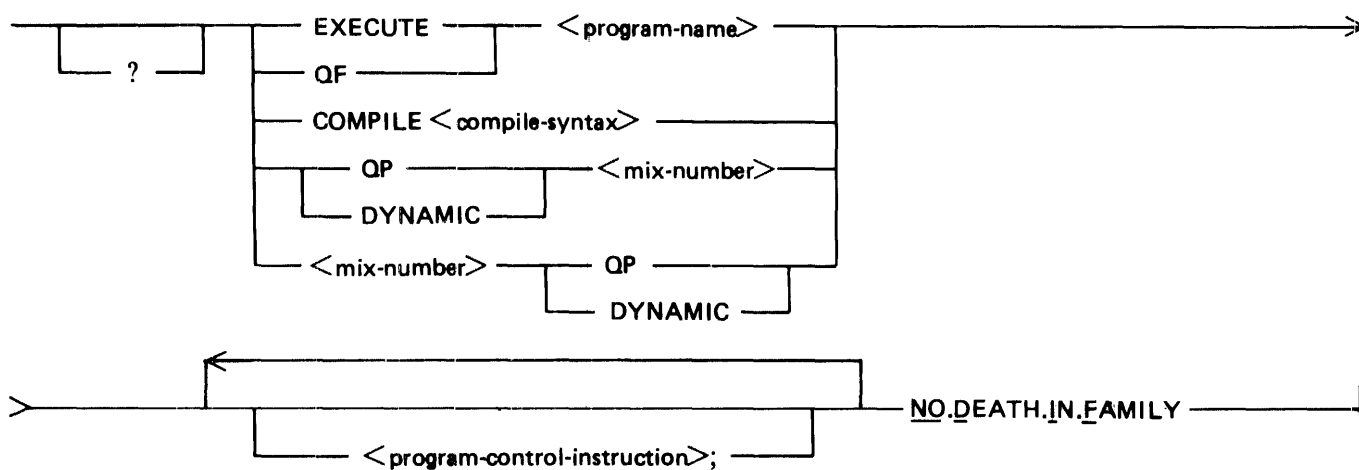
MODIFY A/B PRIORITY 6;

NO.DEATH.IN.FAMILY

The NO.DEATH.IN.FAMILY program control instruction allows a program that is spawned by another program to continue running if the parent (controlling) program terminates.

All programs spawned by a controlling program are terminated by the MCP if the controlling program terminates and the NO.DEATH.IN.FAMILY program control instruction is not specified. If the NO.DEATH.IN.FAMILY program control instruction is specified when the task is executed, the task continues to run even if the parent program terminates. Control is linked to the next higher level, which can ultimately be the MCP. The output is then routed to some other location (for example, to the ODT or the line printer).

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

```
COMPILE A/Y COBOL LI; NO.DEATH.IN.FAMILY;
```

```
DYNAMIC 758 NODIF;
```

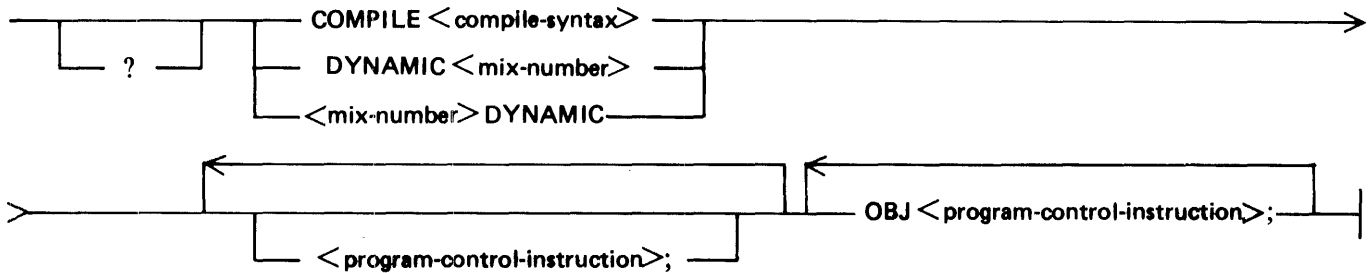
OBJ

The OBJ program control instruction is used as a prefix to another program control instruction and causes that instruction to reference a compiled object program. The OBJ program control instruction must therefore be associated with a compilation.

The OBJ program control instruction is used only after the COMPILE program control instruction or after a DYNAMIC program control instruction that references a compilation. It causes the program control instruction that follows it to refer to the object file of the compilation, instead of referring to the compiler itself.

All modifications specified by program control instructions that are paired with an OBJ program control instruction are queued by the MCP and, following successful compilation, are used to perform an automatic MODIFY program control instruction on the object program. Only those program control instructions that are allowed to follow the MODIFY program control instruction can be used following the OBJ program control instruction.

Syntax:



Semantics:

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed compilation.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow the MODIFY program control instruction.

Examples:

```
COMPILE TEST/PROGRAM COBOL LIBRARY;
FILE CARDS NAME SOURCE/TEST DISK;
OBJ FILE LINE NO HARDWARE LABEL.TYPE = 1;
PRIORITY 5;
OBJ PROCESSOR.PRIORITY 9;
OBJ MEMORY.PRIORITY 5;
MEMORY 25000;
OBJ MEMORY 15000
```

```
DYNAMIC 1234; OBJ PR 6; OBJ FILE CARDS DISK;
```

OVERRIDE

The **OVERRIDE** program control instruction causes the compatibility check normally made between a program and its interpreter to be bypassed.

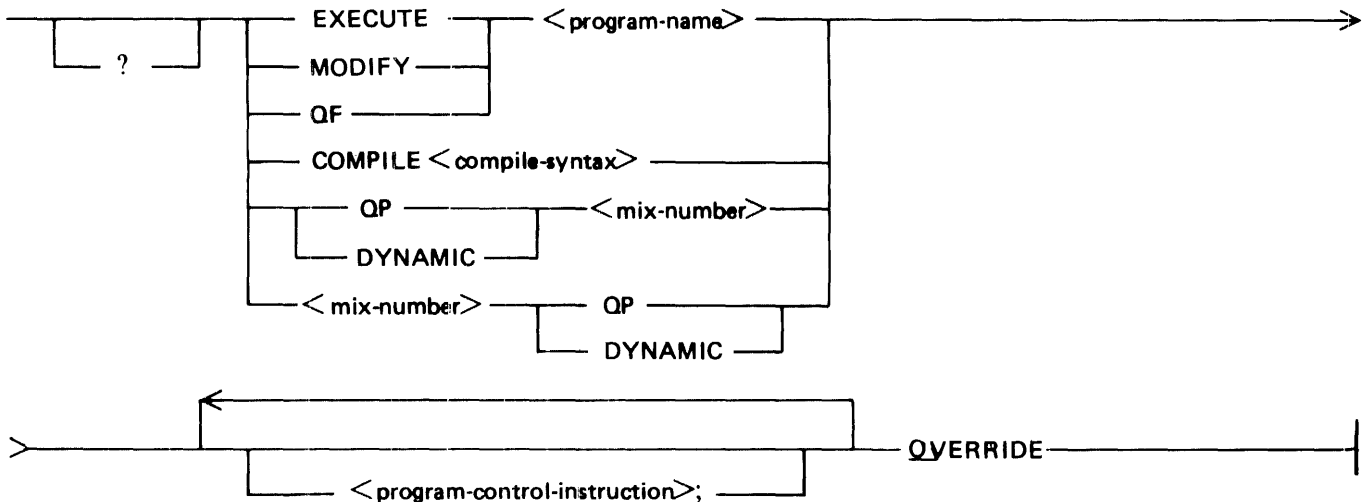
At the beginning of job, the MCP performs a compatibility check of a program and its interpreter, unless the **OVERRIDE** program control instruction is specified. The compatibility check performs the following functions.

- Examines the **HARDWARE.TYPE** field of the interpreter for a value of U (Universal), or matches the type of processor (S or M) on which it is running.
- Compares the **MCP.LEVEL** field of the interpreter with the **LEVEL** field of the MCP.
- Compares the **GISMO.LEVEL** field of the interpreter with the **LEVEL** field of GISMO.
- Compares the **COMPILER.LEVEL** field of the interpreter with the **COMPILER.LEVEL** field of the program.
- Compares the **ARCHITECTURE** (language) field of the interpreter with the **INTERPRETER.FIRST.NAME** field of the program.
- Examines the interpreter to see if it has at least all attributes required by the program.

Using the **OVERRIDE** program control instruction does not bypass the interpreter generation process of the MCP.

The **UNOVERRIDE** program control instruction can be used to reset the **OVERRIDE** program control instruction and causes the compatibility check to be performed.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the **COMPILE** keyword, refer to the **COMPILE** program control instruction in this section.

OVERRIDE

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

EXECUTE A/B; OVERRIDE;

MODIFY TEST OV;

DYNAMIC 275 OV;

PRIORITY

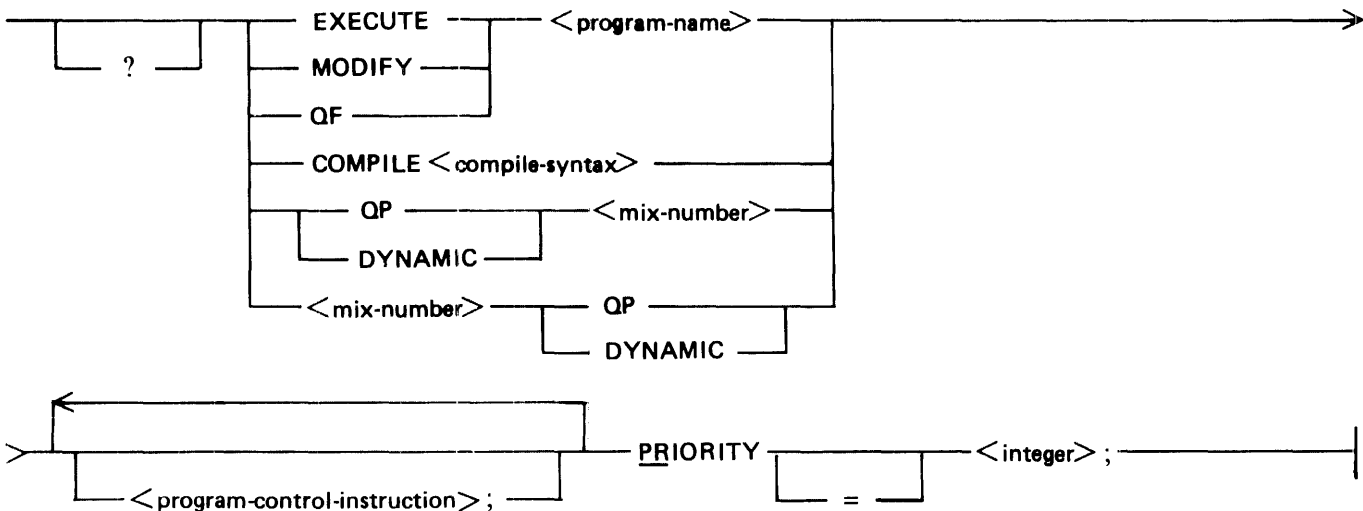
The PRIORITY program control instruction specifies the operational processor priority of a program by setting the PROCESSOR.PRIORITY value.

If the MPRI MCP option is not set, the crashout capabilities normally associated with MEMORY.PRIORITY values of nine or greater are associated with PROCESSOR.PRIORITY instead.

After a program has gone to beginning of job, any queries or changes to the PRIORITY value in the working copy of the program must be accomplished through the MP, PP, or PR system commands described in section 5.

Refer to the MEMORY.PRIORITY and PROCESSOR.PRIORITY program control instructions in this section for a complete description of their use.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 15 inclusive, and specifies the operational priority assigned to the program.

PRIORITY

Examples:

EXECUTE A/B PRIORITY = 10;

COMPILE TEST COBOL LIBRARY PR 9;

PROCESSOR.PRIORITY

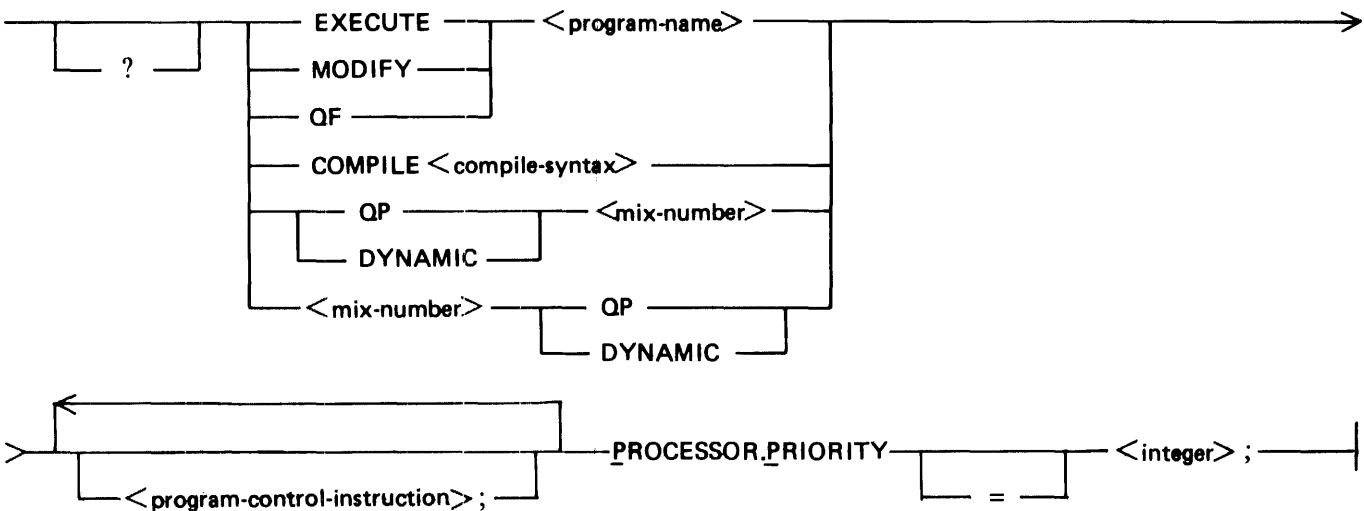
The PROCESSOR.PRIORITY program control instruction specifies the processor usage priority assigned to a program. This instruction is allowed only when the MPRI MCP option is set.

If the value specified by the PROCESSOR.PRIORITY program control instruction is nine or greater, the following actions occur.

- The SCHEDULE.PRIORITY is assigned the same value unless it is explicitly set to some other value using the SCHEDULE.PRIORITY program control instruction.
- The program is not considered by the MCP in determining whether or not the mix limit has been reached. Refer to the ML command in section 5 for a complete description of the mix limit.
- If the MPRI MCP option is not set, the crashout capability invoked when the value of MEMORY.PRIORITY is nine or greater is invoked by the PROCESSOR.PRIORITY program control instruction.

After a program has gone to beginning of job, any queries or changes to the PROCESSOR.PRIORITY value in the working copy of the program must be accomplished through the PP system command described in section 5. Refer to appendix A for a complete description of the MCP memory management functions.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILER keyword, refer to the COMPILER program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

PROCESSOR.PRIORITY

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 15 inclusive, and specifies the processor priority assigned to the program. Zero is the lowest priority and 15 is the highest priority.

Examples:

EXECUTE A/B PROCESSOR.PRIORITY = 6;

COMPILE TEST COBOL LIBRARY PP 5;

PROTECTED

The PROTECTED program control instruction causes the MCP to protect the program against entry of the following system commands described in section 5.

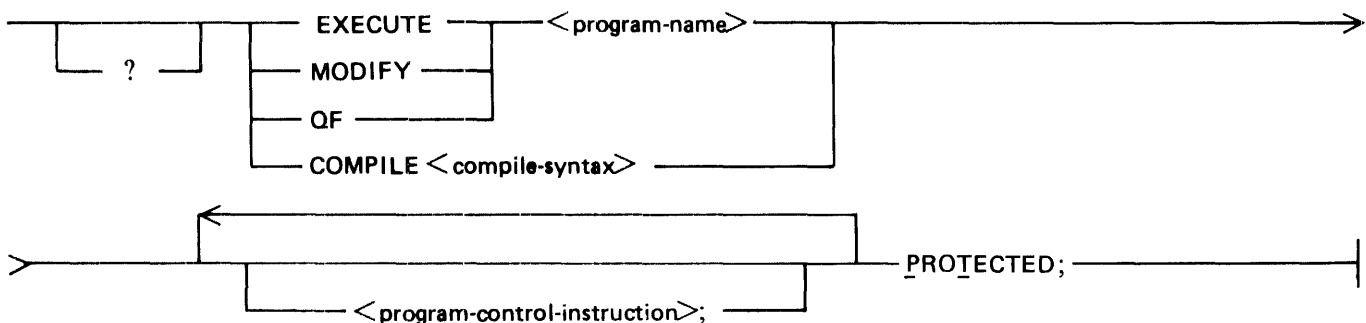
CL
DP
DS
QC
ST
SW

The DP, DS, ST, and SW system commands are allowed to reference a spawned task from the parent remote terminal.

If a protected program reaches an abnormal termination, the MCP automatically unlocks it so that it can be discontinued with the DS or DP system commands.

After a program has gone to beginning of job, any queries or changes to the PROTECTED attribute in the working copy of the program must be accomplished through the LP system command described in section 5.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, or COMPILE program control instruction.

Examples:

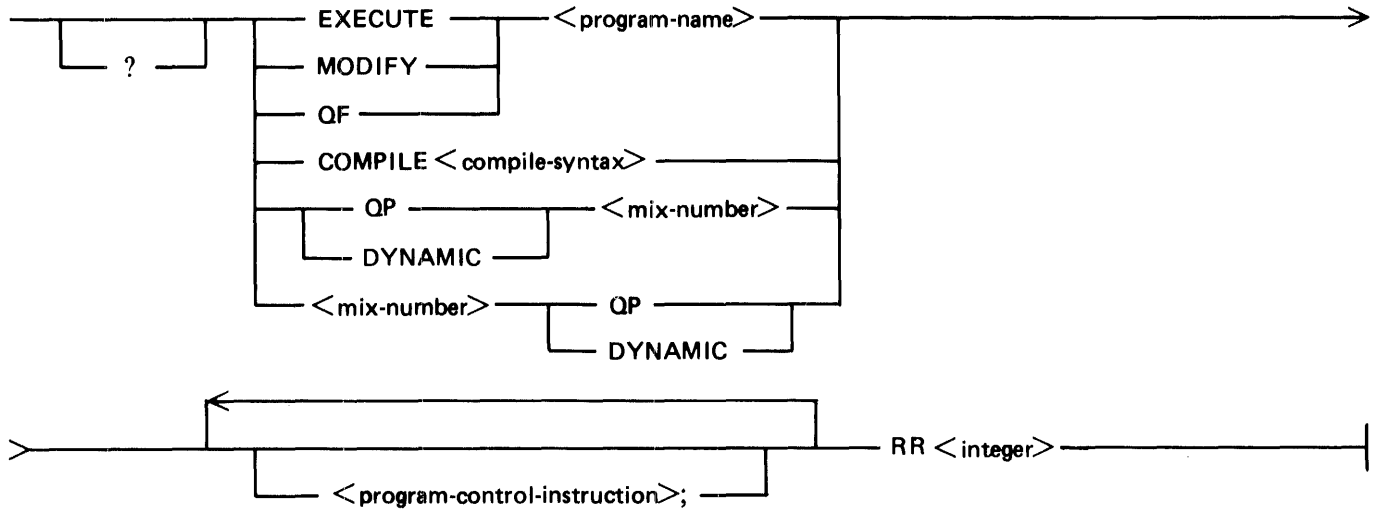
```
EXECUTE REMOTE/UPDATE PROTECTED;
```

```
COMPILE TEST/PROGRAM COBOL74 LIBRARY PT;
```

RR

The RR program control instruction assigns restrictions to a task.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 3 inclusive, and specifies the level of restrictions for the task.

If RR 0 is specified, there are no restrictions applied to the task.

If RR 1 is specified, the USER <usercode>/<password> system command (described in section 5) must be included in ZIP system commands, and direct line printer output is not allowed.

If RR 2 is specified, direct line printer output is not allowed.

If RR 3 is specified, the USER <usercode>/<password> system command (described in section 5) must be included in ZIP system commands, and direct line printer output and card input are not allowed.

Non-zero values of the RR attribute determine which system commands can be zipped from a program. Refer to section 5 for more details.

Examples:

EXECUTE A/B RR 2;

MODIFY TEST/PROGRAM RR 1;

RUN

RUN

The RUN program control instruction causes the MCP to call an object program from a library for subsequent execution. The syntax of the RUN program control instruction is in the B 1000 Systems WFL Language Manual.

START

The START program control instruction causes a WFL job to be initiated. For a complete description of the syntax and semantics of the START program control instruction, refer to the B 1000 Systems WFL Language Manual.

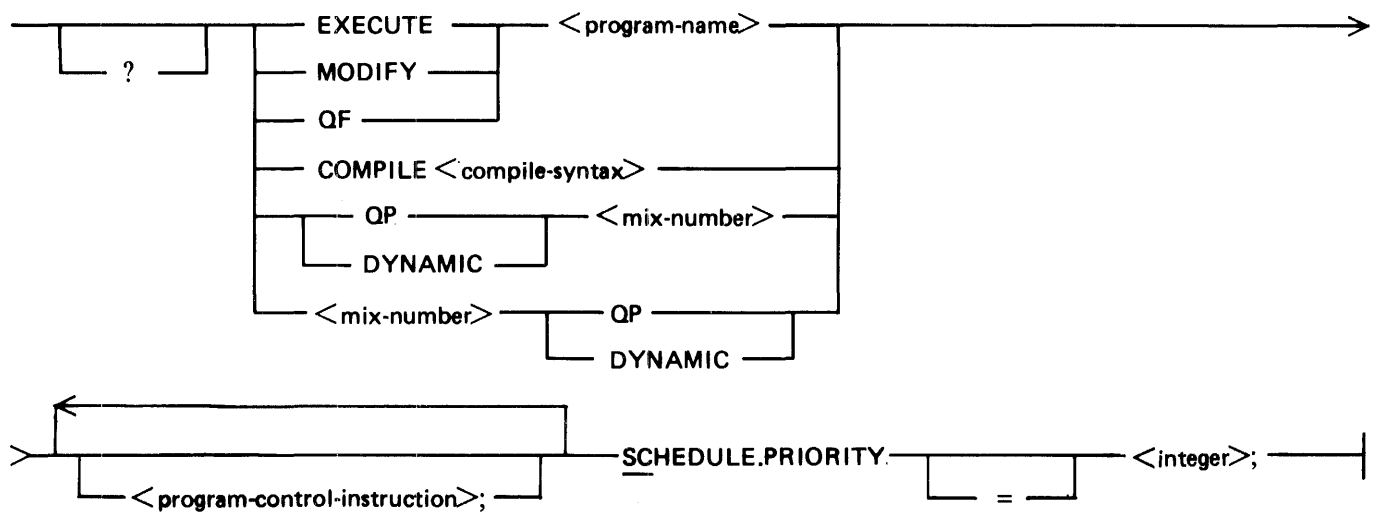
SCHEDULE.PRIORITY

The SCHEDULE.PRIORITY program control instruction assigns priorities to programs in the schedule.

The priority set by the SCHEDULE.PRIORITY program control instruction affects only the schedule priority of a program and is not related to the priorities set by the PROCESSOR.PRIORITY or the MEMORY.PRIORITY program control instructions.

Programs in the active schedule that have the same SCHEDULE.PRIORITY value are scheduled on a first-in, first-out basis.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 14 inclusive, and specifies the priority in which programs in the active schedule are scheduled. Zero is the lowest priority and 14 is the highest priority.

Examples:

EXECUTE A/B SCHEDULE.PRIORITY = 12;

COMPILE TEST/PROGRAM COBOL74 LI SC 7;

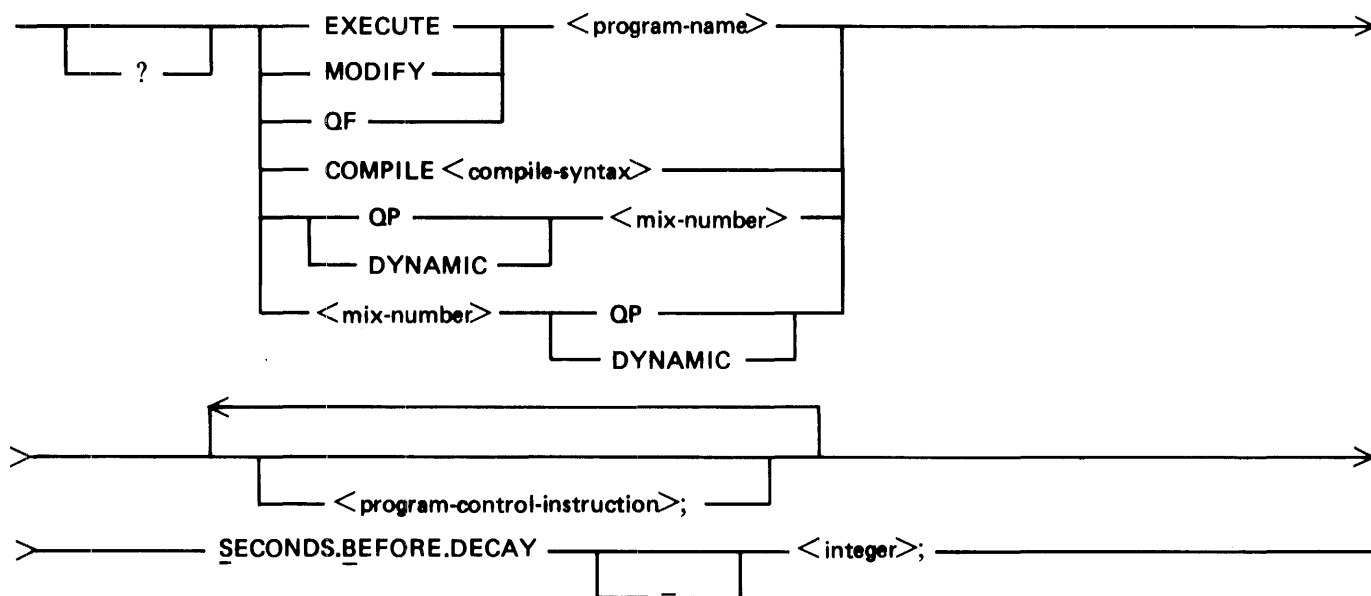
SECONDS.BEFORE.DECAY

The SECONDS.BEFORE.DECAY program control instruction specifies the length of time to protect unreferenced code segments marked as IMPORTANT from being degraded to a lower memory priority. This attribute is valid only when the MPRI MCP option is set.

The SYSTEM/MARK-SEGS system utility program marks specific code segments of a program as IMPORTANT for use with the SECONDS.BEFORE.DECAY program control instruction. Refer to the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the SYSTEM/MARK-SEGS system utility program.

After a program has gone to beginning of job, any queries or changes to the SECONDS.BEFORE.DECAY value in the working copy or the program must be accomplished through the SB system command described in section 5. Refer to appendix A, MCP Memory Management, for further operational details on Priority Memory Management.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 600 inclusive, and specifies the length of time in seconds before an unreferenced code segment marked **IMPORTANT** is reduced (decayed) to a lower memory priority. If `<integer>` is zero, all code segments of a program, whether marked as **IMPORTANT** or not, are treated as not **IMPORTANT**.

Examples:

```
EXECUTE A/B MEMORY.PRIORITY = 15; SECONDS.BEFORE.DECAY = 250;
```

```
EX NDL/HANDLER MP 15; SB 600;
```

STREAM

The STREAM program control instruction informs the MCP of the name of a punched card file that is to be treated as a stream data file.

A stream data file consists of all data cards contained between the stream beginning and the stream end. The stream beginning is identified by the STREAM program control instruction. The stream end is identified by the TERMINATE program control instruction. The file identifier specified in the TERMINATE program control instruction must be the same as on the corresponding STREAM program control instruction to end the stream data file.

The STREAM and TERMINATE program control instructions are similar to DATA CTLDCK and ENDCTL used in pseudo-reader loading, but are more generalized.

When a program reads a stream data file, the EXCEPTION branch is taken any time a card with a question mark (?) or invalid character is read from column one. The MCP replaces column one of that card image with binary zeroes (@00@) prior to passing the card image to the reading program. The program receives the end-of-file exception only when the proper TERMINATE program control instruction has been read.

Syntax:

— | STREAM <file-identifier>; — |
| |
| ? |

Semantics:

file-identifier

The field can contain any valid two-name file identifier and specifies the name of the stream data file.

Example:

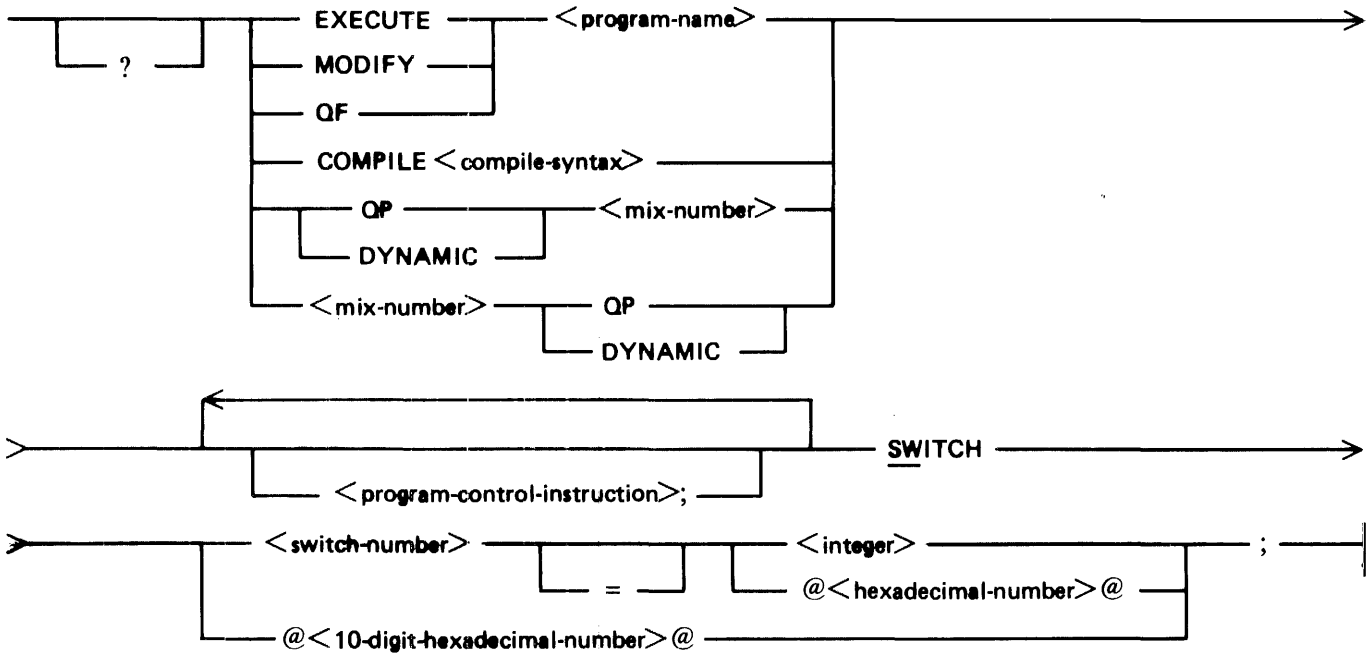
```
?STREAM CARDS;  
?COMPILE TEST WITH COBOL74 LIBRARY;  
?CHARGE 123456;  
?FILE CARDS NAME = SOURCE/TEST DISK;  
?EXECUTE TEST CHARGE 123456;  
?DATA CARDIN;  
.  
.(data cards)  
.  
?END CARDIN;  
?TERMINATE CARDS;
```

SWITCH

The SWITCH program control instruction sets the program switches used by a program.

To modify or query the switches after that program has gone to beginning of job, refer to the SW and TS system commands, respectively, described in section 5.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

switch-number

This field can contain any integer in the range 0 to 9 inclusive, and specifies the switch to be assigned a value.

SWITCH

integer

This field can contain any integer in the range 0 to 9 inclusive, and specifies the value of the switch.

hexadecimal-number

This field can contain any hexadecimal number in the range 0 to F inclusive, and specifies the value of the switch.

10-digit-hexadecimal-number

This field can contain any 10-digit hexadecimal number in the range of 0000000000 to FFFFFFFF inclusive, and the ordinal position of each hexadecimal digit corresponds to its respective switch. For example, specifying SWITCH = @0070000000@ causes switch 2 to be set to 7.

Examples:

EXECUTE A/B SWITCH 0 = 5; SWITCH 1 = 3;

COMPILE COBOL74 SW 1 = 1;

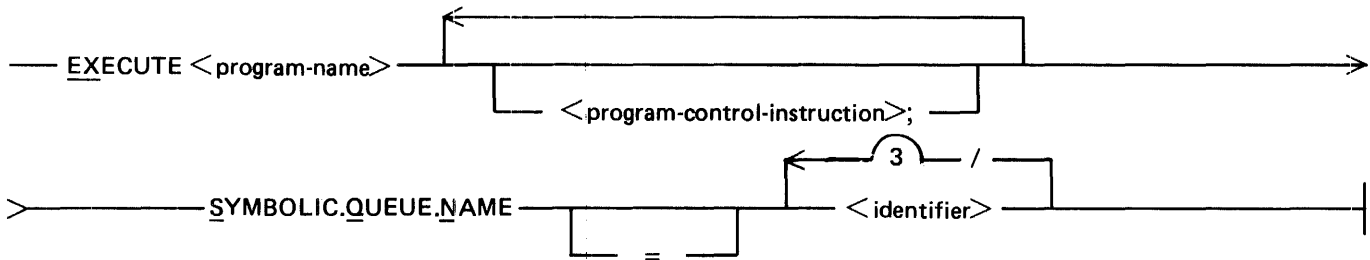
EXECUTE TEST/PROGRAM SW = @00013F0000@;

SYMBOLIC.QUEUE.NAME

The SYMBOLIC.QUEUE.NAME program control instruction initializes the SYMBOLIC.QUEUE.NAME of the INITIAL INPUT CD of a COBOL 74 program.

The SYMBOLIC.QUEUE.NAME syntax is used by an MCS. When an MCS executes a COBOL 74 program that uses a queue created previously by the MCS, the MCS initializes the SYMBOLIC.QUEUE.NAME in the INITIAL INPUT CD using the SYMBOLIC.QUEUE.NAME syntax. The SYMBOLIC.QUEUE.NAME value can be set only on a ZIP; it cannot be modified. A COBOL 74 program with an INITIAL INPUT CD that is not executed with a SYMBOLIC.QUEUE.NAME value set has blanks (spaces) put into the INITIAL INPUT CD queue name.

Syntax:



Semantics:

program-name

This field can contain any valid COBOL 74 program name.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow the EXECUTE program control instruction.

identifier

This field is an identifier containing from 1 to 12 characters. When the queue is created, only the first 10 characters of the first `<identifier>` are used by the COBOL 74 program as the queue name. That is, the COBOL 74 program must ensure that the last 38 characters of the SYMBOLIC.QUEUE.NAME attribute are blank.

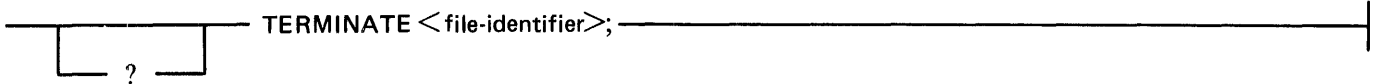
TERMINATE

TERMINATE

The TERMINATE program control instruction informs the MCP that the stream data file has reached the end of file.

Refer to the STREAM program control instruction for additional information concerning the use of the TERMINATE program control instruction.

Syntax:

 TERMINATE <file-identifier>;

Semantics:

file-identifier

This field can contain any valid one- or two-name file identifier and specifies the same name as specified in the corresponding STREAM program control instruction.

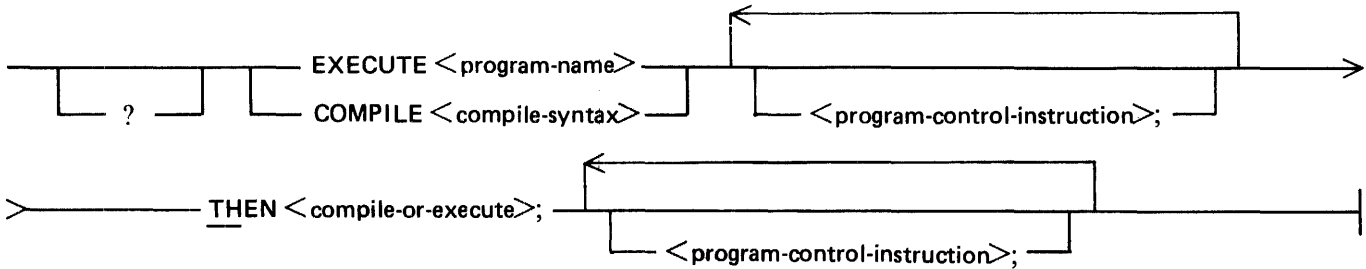
Example:

```
?STREAM CARDS;  
?COMPILE TEST WITH RPG LIBRARY;  
?CHARGE 123456;  
?FILE CARD NAME = SOURCE/TEST DISK;  
?EXECUTE TEST CHARGE 123456;  
?DATA CARDIN;  
  .  
  .(data cards)  
  .  
?END CARDIN;  
?TERMINATE CARDS;
```

THEN

The THEN program control instruction conditionally schedules the execution of a program in relation to another program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

compile-or-execute

This field can contain any valid COMPILE or EXECUTE program control instruction.

Examples:

```
EXECUTE A/B THEN COMPILE BETA COBOL74 SYNTAX; UNCONDITIONAL;
```

```
COMPILE TEST/PROGRAM COBOL74 LI TH EX TEST;
```

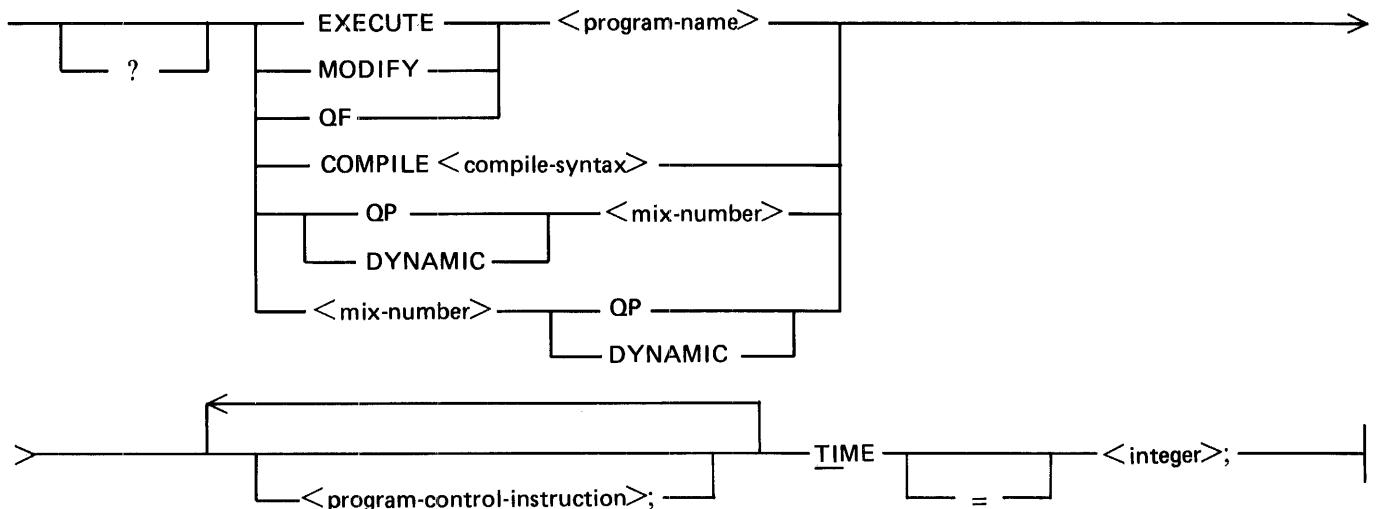
TIME

The TIME program control instruction specifies the maximum allowable processor time a program can accumulate.

If a processor time limit is set for a program with the TIME program control instruction and the program exceeds the time limit, it is discontinued by the MCP and the following message is displayed.

EXCEEDED MAXIMUM RUN TIME ALLOWED

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 1 to 27,962 inclusive, and specifies the amount of processor time (not elapsed time) in minutes that a program can accumulate before being discontinued.

Examples:

EXECUTE A/B TIME = 5;

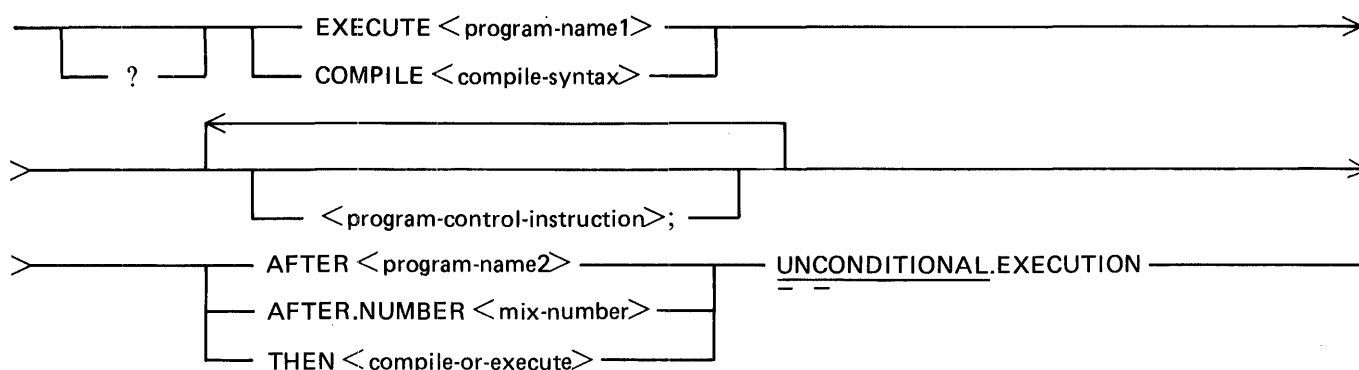
COMPILE TEST/PROGRAM COBOL74 LIBRARY TI 12;

UNCONDITIONAL.EXECUTION

The UNCONDITIONAL.EXECUTION program control instruction is used in conjunction with the AFTER, AFTER.NUMBER, and THEN program control instructions and forces a program to beginning of job regardless of the outcome of its predecessor.

If the UNCONDITIONAL.EXECUTION program control instruction is not specified for a successor program and the predecessor abnormally terminates, or is a compilation that terminates with syntax errors, the MCP places that program in the WAITING SCHEDULE and waits for an RS or FS system command described in section 5.

Syntax:



Semantics:

program-name1

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE or COMPILE instruction.

program-name2

This field can contain any valid B 1000 program name and specifies the name of an executing or scheduled to be executed program. The program specified by <program-name2> must terminate before the execution specified by <program-name1> or the compile specified by <compile-syntax> is performed.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program. The program specified by <mix-number> must terminate before either the execution specified by <program-name1> is done or the compile specified by <compile-syntax> is performed.

compile-or-execute

This field can contain any valid COMPILE or EXECUTE program control instruction.

B 1000 System Software Operation Guide, Volume 1
Program Control Instructions

Examples:

EXECUTE A/B AFTER C/D; UNCONDITIONAL.EXECUTION;

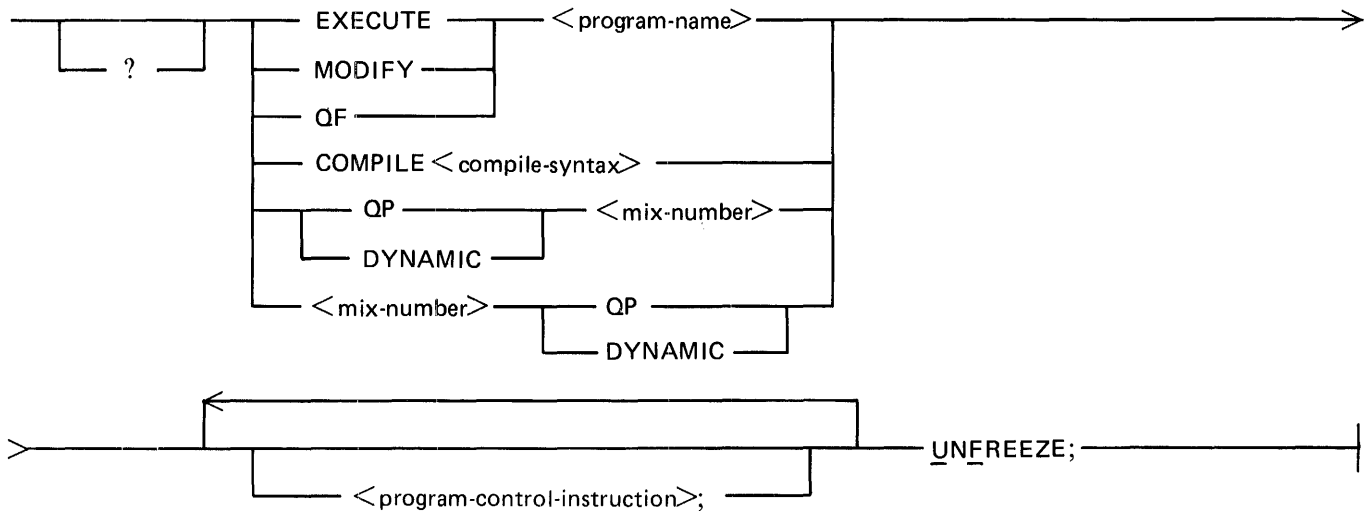
EX A/B AF C/D; UNCONDITIONAL;

COMPILE TEST/PROGRAM COBOL74 LIBRARY; THEN EXECUTE TESTER; UC;

UNFREEZE

The UNFREEZE program control instruction attribute removes the FREEZE condition from a program and permits the base-to-limit area of a program to be rolled out to disk when the program is in an interrupted state. The base-to-limit area of a program contains the data for the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

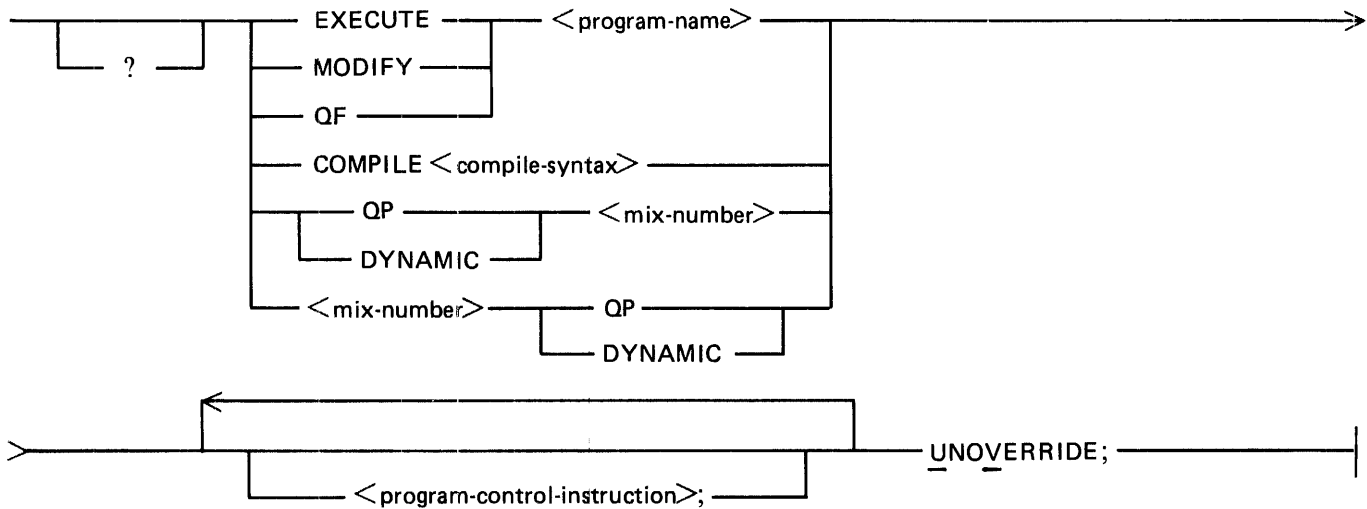
```
EXECUTE A/B UNFREEZE;
```

```
COMPILE TEST/PROGRAM COBOL74 LIBRARY UF;
```

UNOVERRIDE

The UNOVERRIDE program control instruction resets the compatibility check bypass caused by specifying the OVERRIDE program control instruction. Refer to the OVERRIDE program control instruction in this section for a description of the interpreter compatibility checking functions.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

Examples:

```
EXECUTE A/B UNOVERRIDE;
```

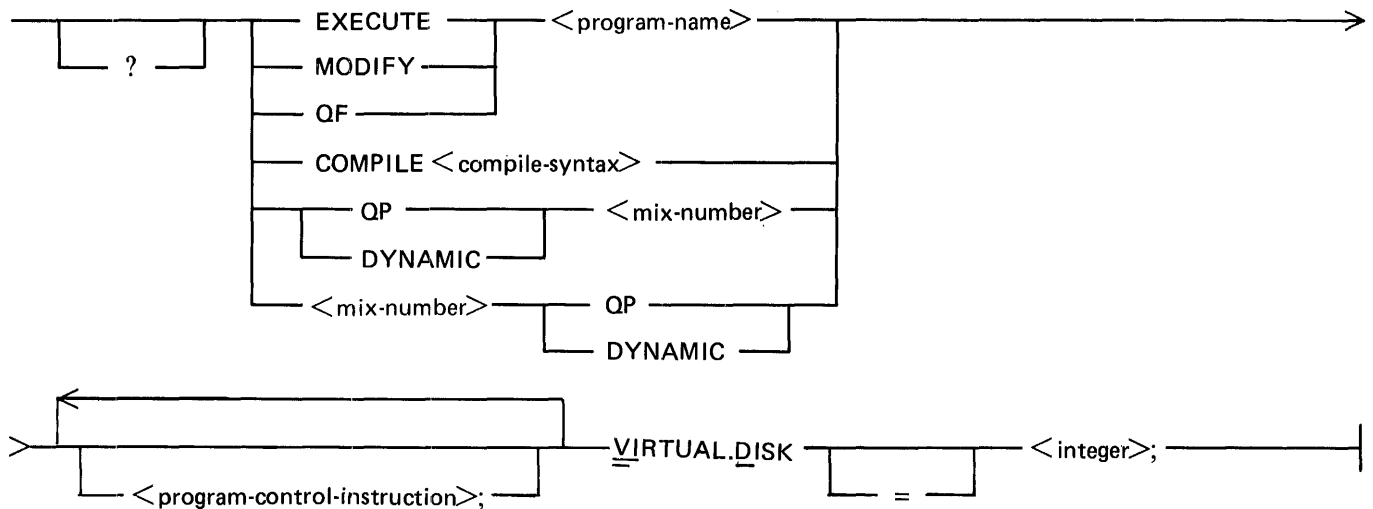
```
MODIFY TEST/PROGRAM UV;
```

VIRTUAL.DISK

The VIRTUAL.DISK program control instruction changes the number of disk segments assigned to hold non-memory-resident data overlays during program execution. Virtual disk is the secondary storage area for overlayable data and dynamic memory is the primary storage area.

If the value provided by the VIRTUAL.DISK program control instruction is zero and the program requires disk space for data overlays, the MCP assigns a default size of 1000 disk segments to the program.

Syntax:



Semantics:

program-name

This field can contain any valid B 1000 program name.

compile-syntax

For a complete description of the syntax following the COMPILE keyword, refer to the COMPILE program control instruction in this section.

mix-number

This field can contain any mix number of an executing or scheduled-to-be-executed program.

program-control-instruction

This field can contain any valid B 1000 program control instruction described in this section that is allowed to follow either the EXECUTE, MODIFY, QF, COMPILE, QP, or DYNAMIC program control instruction.

integer

This field can contain any integer in the range 0 to 16,777,215 inclusive, and specifies the number of disk segments to be used for data overlays.

Examples:

EXECUTE A/B VIRTUAL.DISK = 4000;

COMPILE TEST/PROGRAM UPL LIBRARY VI 6000;

EX TESTER VD 2000;

SECTION 5

SYSTEM COMMANDS

This section contains, in alphabetic order, each system command and its description. Descriptions include railroad syntax diagrams.

COMMAND ENTRY

All system commands may be entered through the system ODT. System commands may also be entered through two types of remote stations, a remote ODT and a workstation.

A remote ODT is a station whose MCS appends a restriction attribute of zero ($RR = 0$) to commands entered through it. All the system commands except HALT and NET are accepted through a remote ODT.

A workstation is a station whose MCS appends $RR = 1, 2,$ or 3 to commands. This restricts commands accepted through a workstation to a subset that is specified by the RR value.

SYNTAX DIAGRAMS

In this section, each syntax diagram is labeled in one of four ways: Syntax, Remote ODT syntax, Workstation syntax, System ODT Syntax.

Commands with syntax diagrams labeled syntax are accepted from all three station types – system ODT, remote ODT, workstation.

Commands with syntax diagrams labeled Remote ODT Syntax are accepted from the system ODT or from a remote ODT.

Commands with syntax diagrams labeled Workstation syntax are accepted from a workstation. These diagrams include usercodes, but the usercode may be omitted when the command is entered by a signed-on user because the SMCS and CANDE programs prefix the workstation's `<usercode>/<password>` to the command.

Commands with syntax diagrams labeled System ODT Syntax (the HALT and NET commands only) are valid for entry through the system ODT only.

Multiple system commands are allowed if they are separated by a semicolon (;) or a blank character. There is no valid separator for the AC command.

AB (AUTOBACKUP)

The AB system command reserves line printer devices and the SYSTEM/BACKUP program for the MCP autoprint mechanism. The autoprint mechanism retrieves printer backup files from disk and prints them automatically, directing the output to the designated line printer. Up to four copies of the SYSTEM/BACKUP program can be specified for execution and remain in the mix, as long as there are backup files to be printed.

When the autoprnt mechanism is invoked, a designated number of SYSTEM/BACKUP programs are executed. The names of candidate printer backup disk files are entered in a queue file named AUTO-PRINT. Each SYSTEM/BACKUP program initiated by the autoprnt mechanism reads entries from this queue and prints the file specified. Printing is done as though a PB system command had been entered with no options specified. Multiple copies can be printed by specifying a non-zero value for the REPETITIONS file attribute. Refer to the FILE program control instruction in section 4 for a complete description of the REPETITIONS file attribute.

If the SYSTEM/BACKUP program has program switch 3 set to 0, the printer backup disk files are removed after being printed.

The SYSTEM/BACKUP programs can be made to remain in the mix as long as there are backup files to be printed. The SYSTEM/BACKUP program terminates when the AUTOPRINT queue is empty and program switch 7 is set to a non-zero value. If the AUTOPRINT queue is empty for a period of five minutes, the SYSTEM/BACKUP program terminates.

Refer to the SYSTEM/BACKUP program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the autoprnt mechanism.

The SYSTEM/BACKUP programs initiated by the autoprnt mechanism use only the line printers reserved by the AB system command. Backup print disk files from other programs in the mix, including other SYSTEM/BACKUP programs not initiated by the autoprnt mechanism, cannot use any line printer reserved by the autoprnt mechanism.

The AB system command has the following forms.

AB

Entering the AB system command without any options causes the current autoprnt value and the unit mnemonics of all line printers reserved for the autoprnt mechanism to be displayed.

AB <integer>

Entering the AB system command followed by <integer> causes <integer> number of copies of the SYSTEM/BACKUP program to be executed. If <integer> equals 0, the AUTOPRINT queue is purged, no further backup printer disk file names are allowed to be entered, and currently running SYSTEM/BACKUP programs complete the printing of their respective backup print disk files and go to end of job.

AB + <unit-mnemonic>

Entering the AB system command with + <unit-mnemonic> reserves the specified line printer device for the autoprnt mechanism. The + key symbol is optional.

AB -<unit-mnemonic>

Entering the AB system command with -<unit-mnemonic> releases the specified line printer device from the autoprnt mechanism.

NOTE

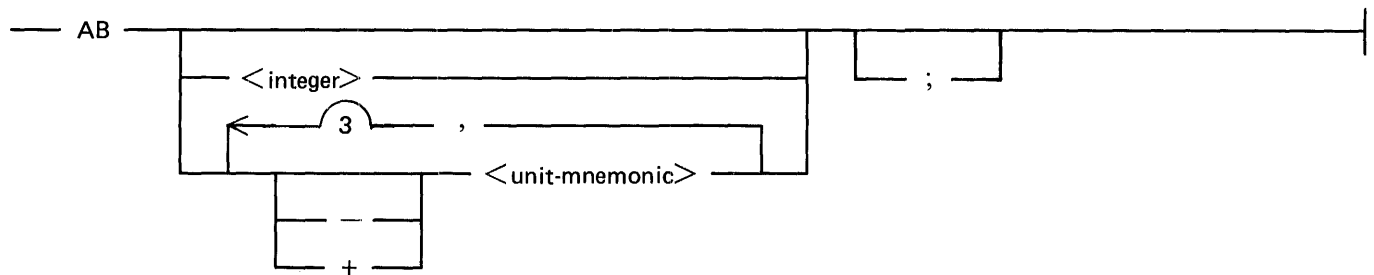
It is the responsibility of the operator to match the number of line printers reserved for the autoprnt mechanism with the number of copies of the SYSTEM/BACKUP program specified by AB <integer>. It is not desirable, for example, to specify a multiple number of SYSTEM/BACKUP programs while reserving a single line printer for the autoprnt mechanism. Likewise, there is no advantage in reserving more line printers than the number of SYSTEM/BACKUP programs specified.

AB

The autoprnt mechanism can be activated in one of the following four ways.

1. Entering AB <integer> sets up the autoprnt values and reserves the line printers. Also, if any of the SYSTEM/BACKUP programs are discontinued with the DS or DP system commands, entering AB <integer> reactivates the autoprnt mechanism.
2. If the autoprnt mechanism was invoked and a clear/start operation is required, the autoprnt parameters are retained through the clear/start operation and the autoprnt mechanism is automatically begun after the clear/start operation.
3. If a backup disk file is closed and is a candidate for the autoprnt mechanism, and there are no copies of the SYSTEM/BACKUP program in use by the autoprnt mechanism, the MCP executes the required number of SYSTEM/BACKUP programs.
4. If the backup designated disk (DL system command) is made ready and the AB value is non-zero, the MCP activates the autoprnt mechanism. Only backup files residing on the default backup user disk are printed by the autoprnt mechanism.

Syntax:



Semantics:

integer

This field, which must contain an integer in the range 0 to 4 inclusive, specifies the number of SYSTEM/BACKUP programs to be used.

unit-mnemonic

This field, which must contain a valid line printer unit mnemonic, for example, LPA, LPB, LPC, or LPD. Specifying <unit-mnemonic> causes the line printer to be reserved for the autobackup mechanism.

-

The - key symbol releases the line printer from the autobackup mechanism.

+

The + key symbol is optional and is equivalent to specifying <unit-mnemonic> without the + or - key symbols.

Examples:

Command	Response
AB;	AUTO BACKUP - AB = 0 NO PRINTERS RESERVED
AB 2;	NO BACKUP FILES ON DISK
AB LPA;	LPA RESERVED FOR AUTO BACKUP.
AB;	AUTO BACKUP - AB = 2 AND LPA RESERVED
AB LPB, -LPA;	LPB RESERVED FOR AUTO BACKUP. LPA NO LONGER RESERVED FOR AUTO BACKUP.
AB 0;AB+LPA,-LPB;	AB INTEGER = 0 LPA RESERVED FOR AUTO BACKUP. LPB NO LONGER RESERVED FOR AUTO BACKUP.

AC (Response to Accept Message)

The AC system command is a response to an accept message requested by an object program through the MCP. It functions similarly as the AX system command; however, it cannot be delimited with a semicolon (;) character. If a semicolon (;) character follows the AC keyword, it is treated as if it were a part of the response.

The AC system command has an unsolicited ODT feature. The operator can enter up to ten AC system commands for a given program prior to the display of the actual accept message. The AC system commands must be entered in the order they are to be used, because the queue has a first-in, first-out structure.

The queue is automatically cleared when the program goes to end of job or when the program is discontinued with a DS or DP system command.

Syntax:

— <mix-number> AC <message> —|

Semantics:

mix-number

This field, which must be a mix number of a currently executing program, specifies the program to which the <message> is written.

message

This field, which must contain a group of alphanumeric characters, specifies the text that is written to the program. <message> starts in the first position after the AC keyword. If <message> is shorter than the receiving field in the program, <message> is padded on the right with blank characters. If <message> is longer than the receiving field in the program, <message> is truncated on the right.

Examples:

1234ACLIST SOURCE/FILE A

8900ACDPA

632AC CHECK VOID IF OVER 500 DOLLARS

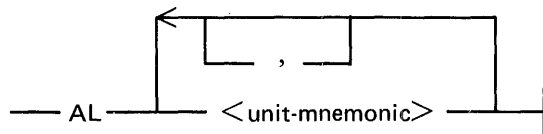
ADD

The ADD system command executes the SYSTEM/COPY program. Refer to the SYSTEM/COPY program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the ADD system command.

AL (Align Printer)

The AL system command aligns a printer device such that the print line at the top of a printer form appears in the desired location. The AL system command causes a complete line of repeated digits (from 0 to 9) to be written to the specified device or devices. If the position of the print line is not satisfactory, the operator can appropriately adjust the printer form and repeat the AL system command as necessary before allowing a program to actually open and use the printer.

Syntax:



Semantics:

unit-mnemonic

This field, which must contain a valid printer unit mnemonic, specifies the unit on which the printer line is to be aligned.

Examples:

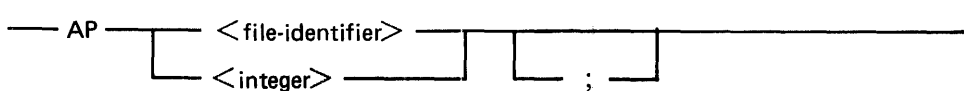
AL LPA

AL LPC LPB

AP (Auto Print)

The AP system command inserts printer backup files into the autoprint queue if the autoprint mechanism is invoked. Usercodes and file security are applied to <file-identifier> and <integer> as required. If <integer> is specified without a usercode, the MCP supplies BACKUP/PRT<integer> for the first part and second part of the file name, and supplies the user disk designated as backup for the family name.

Remote ODT syntax:



Workstation syntax:



Semantics:

file-identifier

This field, which must contain a valid printer backup disk file, specifies the printer backup disk file to be printed.

integer

This field must contain an integer. If the USER command precedes <integer>, the MCP supplies the usercode as the first name and user disk name if a default user disk is applicable. If the USER command does not precede <integer>, the MCP supplies BACKUP/PRT<integer> as the first part and second part of the file name.

Examples:

Command	Results
AP A/B;	Prints the file labeled A/B.
AP 152;	Prints the file labeled BACKUP/PRT152.
US X/Y AP 173;	Prints the file labeled (X)/PRT173.

AT

The AT system command causes <message> to be routed to a remote BNA host system. This instruction is valid from the ODT, remote terminals, or card readers. Chaining of AT system commands is not allowed. For example, specifying AT HOST2 AT HOSTX M is invalid. Responses to AT system commands are identified with the following prefix: FROM <hostname>.

<message> is not scanned or interpreted at the local host; it is in the syntax of the remote host system. An exception to this is when the AT system command is entered from a card reader (or pseudo reader); if the text following <hostname> is the STREAM program control instruction, then a file name is expected following the STREAM program control instruction, and the card file is transferred to the remote host for execution.

Syntax:

— AT <hostname><message>; —|

Semantics:

hostname

This field, which must contain a valid BNA host name in the BNA network, specifies the name of the BNA host destination for <message>.

message

This field must contain a valid message to a remote BNA host. <message> is not syntaxed on the local BNA system and is routed to the remote BNA host for syntax checking and execution.

Examples:

AT HUB COMPILE TEST RPG LI; FILE CARDS NAME SOURCE/TEST DISK;

AT SLAVE1 DF COBOL74/=;

AT B6800 RUN SYSTEM/DUMPALL ("TEACH");

?AT HOSTX STREAM TESTFILE;

.(control cards)

?TERMINATE TESTFILE;

AX (Response to Accept Message)

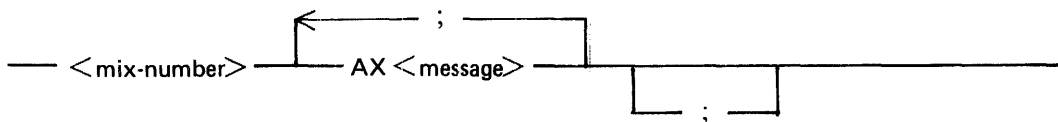
The AX system command is a response to an accept message requested by an object program through the MCP. It functions exactly like the AC system command; however, the semicolon (;) character can be used to delimit multiple AX system commands.

If the semicolon (;) character is encountered immediately after the AX system command, the MCP fills the message area in the requesting program with blank characters.

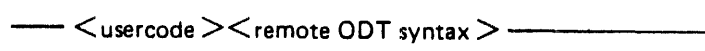
The AX system command has an unsolicited console feature in that the operator can enter any number of AX system commands for a given program prior to the time the actual accept message is displayed. The AX commands must be entered in the order they are to be used, because the queue has a first-in, first-out (FIFO) structure.

The queue is automatically cleared when the program goes to end of job or when the program is discontinued with a DS or DP system command.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must be a mix number of a currently executing program, specifies the program to which <message> is written.

message

This field, which must contain a group of alphanumeric characters, specifies the text that is written to the program. <message> starts in the first position after the AX keyword. If <message> is shorter than the receiving field in the program, <message> is padded on the right with blank characters. If <message> is longer than the receiving field in the program, <message> is truncated on the right.

;

The semicolon (;) character terminates the response to the accept message.

Examples:

```
1234AXLIST SOURCE/FILE A;
```

```
8900AXDPA;
```

```
632AX CHECK VOID IF OVER 500 DOLLARS;
```

BB (Backup Blocks per Area)

The BB system command specifies the number of blocks to assign each area of a printer or punch backup disk file.

The value of the backup blocks per area is set to 200 by the COLDSTART/TAPE or COLDSTART/DISK programs and, if <integer> is less than 5, a value of 200 is assigned by default.

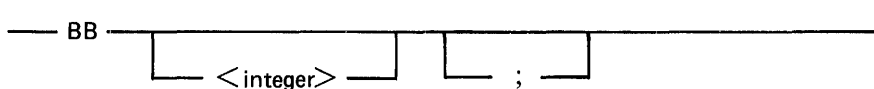
If <integer> is not specified in the BB system command, the MCP displays the current setting of the backup blocks per area.

When an output printer or punch file is opened on disk, its file attributes are set by the MCP in the following ways.

- The value of the RECORD.SIZE file attribute is increased by 1 for punch backup files and 2 for printer backup files.
- The value of the RECORDS.PER.BLOCK file attribute is optimized by the MCP to realize optimum use of disk space.
- The value of the BLOCKS.PER.AREA file attribute is assigned the value of the BLOCKS.PER.AREA file attribute declared either in the program or the value of <integer> specified in the BB system command, whichever is larger.
- The value of the AREAS file attribute is assigned the value of the AREAS file attribute declared in the program or 25, whichever is larger.

It is possible to declare, by way of the FILE program attribute, large file sizes (BLOCKS.PER.AREA and AREAS file attributes) for specific backup files that are known to be larger than normal, while maintaining a system-wide default for all other backup files.

Remote ODT syntax:



Semantics:

integer

This field, which must contain an integer, specifies the default number of blocks per area to assign to each area of a printer or punch backup file. The MCP does not permit a value of less than 5 to be specified.

Examples:

Command	Response
BB	PBD BLOCKS PER AREA = 200.
BB 350	# PBD BLOCKS.AREA CHANGED TO 350.

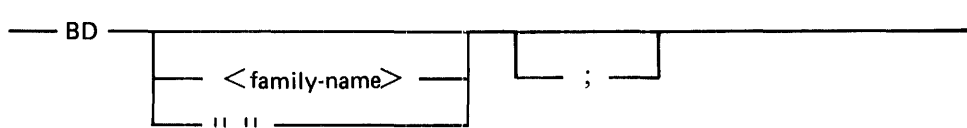
BD (Backup Designate)

The BD system command designates a specific user disk for backup files and dump files.

When the BD message is entered without <family-name>, the MCP causes the current setting of the default backup disk to be displayed.

If a null string (" ") is entered as <family-name>, the default backup disk is changed to the SYSTEM disk.

Remote ODT syntax:



Semantics:

family-name

This field, which must contain a disk identifier, specifies the name of the default disk for backup print and punch files.

" "

The blank character enclosed by the double quotation mark (") characters causes the default backup designate to be the SYSTEM disk.

Examples:

Command	Response
BD;	PBD DESIGNATION = SYSTEM DIRECTORY
BD USER;	DEFAULT PBD DESIGNATION CHANGED TO "USER"
BD " ";	DEFAULT PBD DESIGNATION CHANGED TO SYSTEM

NOTE

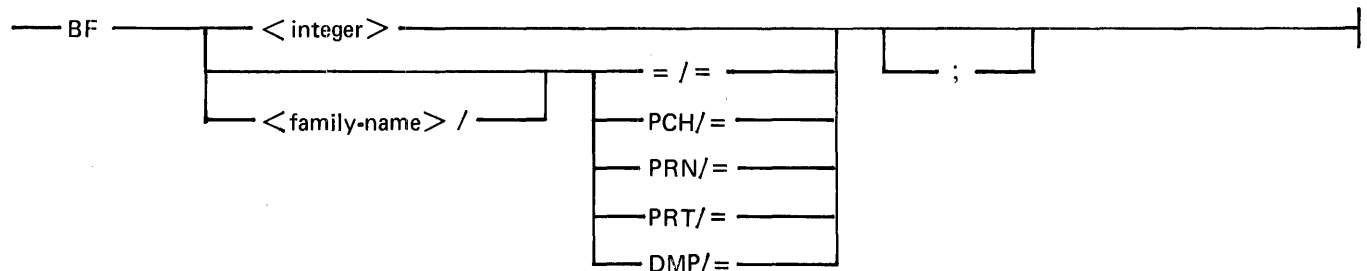
The BD system command will not be supported after the Mark 12.0 release. It has been replaced by the DL system command. The DL system command is described later in this section.

BF (Display Backup Files)

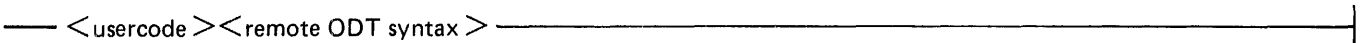
The BF system command lists disk backup files and dump files on the ODT. Entering BF PRT/=, BF PCH/=, and BF DMP/= causes all printer backup, punch backup, and dump files on disk, respectively, to be listed.

If the USER command does not precede the BF system command, the backup printer files listed have the name BACKUP/PRT<integer>, the backup punch files listed have the name BACKUP/PCH<integer>, and the dump files listed have the name DUMPFIL/<integer>. These names are MCP-generated, default names. If the USER command precedes the BF system command, the MCP uses the usercode supplied and lists the printer backup and punch backup files with the MCP-generated, default names of <usercode>/PRT<integer> and <usercode>/PCH<integer>, respectively. If a default disk identifier for the usercode is not the SYSTEM disk, then that user disk is searched for the backup printer and punch files.

Remote ODT syntax:



Workstation syntax:



Semantics:

DMP/=

The DMP/= keysymbol causes the dump files with the MCP-generated default name of DUMPFIL/<integer> to be listed on the ODT.

PCH/=

The PCH/= keysymbol causes the backup punch files with the MCP-generated default name of BACKUP/PCH<integer> to be listed on the ODT.

PRN/=

The PRN/= keysymbol causes the backup printer files with the MCP-generated default name of BACKUP/PRT<integer> to be listed on the ODT.

PRT/=

The PRT/= keysymbol causes the backup printer files with the MCP-generated default name of BACKUP/PRT<integer> to be listed on the ODT.

integer

This field, which must be a 4-digit integer, specifies the backup number of the backup file.

family-name

This field, which must contain a disk identifier, specifies the name of the user disk on which the backup file is located.

=/=

The =/= key symbol causes the backup printer, backup punch, and dump files with the MCP-generated default names of BACKUP/PRT<integer>, BACKUP/PCH<integer>, and DUMPFILE/<integer> to be listed on the ODT.

Examples:

Command	Response
BF =/=	BF = BACKUP/PRT9176 BF = BACKUP/PRT9260 BF = BACKUP/PRT9277 BF = BACKUP/PCH9355 BF = BACKUP/PCH9360 BF = BACKUP/PCH9362 BF = DUMPFILE/9300 BF = DUMPFILE/9417 BF = DUMPFILE/9477 END BF.
BF PRT/=	BF = BACKUP/PRT9176 BF = BACKUP/PRT9260 BF = BACKUP/PRT9277 END BF.
BF PCH/=	BF = BACKUP/PCH9355 BF = BACKUP/PCH9360 BF = BACKUP/PCH9362 END BF.
BF DMP/=	BF = DUMPFILE/9300 BF = DUMPFILE/9417 BF = DUMPFILE/9477 END BF.
BF USER/PRT/=	BF = USER/BACKUP/PRT0020 BF = USER/BACKUP/PRT0100 BF = USER/BACKUP/PRT0277 END BF.
BF PRT/9176	BF = BACKUP/PRT9176 END BF.
USER A/B BF =/=;	BF = USERPACK/(A)/PRT1276 BF = USERPACK/(A)/PCH2789 END BF.

CC (Control Card)

The CC system command specifies that the system command immediately following it is to use the Control Card syntax. For more information, refer to the WFL System Command or the WFL System Option.

Remote ODT syntax:

— CC —|

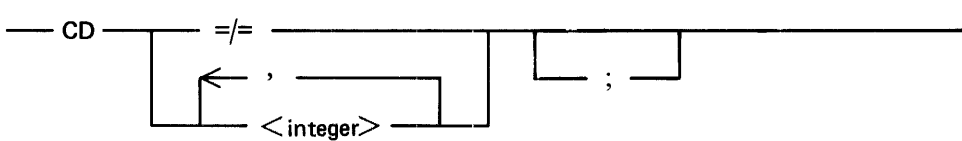
CD (List Card Decks in Pseudo Readers)

The CD system command lists the pseudo card files and their file numbers that have been previously placed on disk by the SYSTEM/LDCONTRL program.

The MCP displays the number of each pseudo card file and the first 50 characters of the first card in the deck.

If a psuedo card file is in use, its name and the program using it is displayed.

Syntax:



Semantics:

=/=

The =/= keysymbol causes the MCP to display all the pseudo card files.

integer

This field, which must contain an integer in the range 0 to 16,777,215 inclusive, causes the MCP to search and display the pseudo card file with the name DECK/<integer>.

Examples:

Command	Response
CD =/=	DECK #1 = ?QU SMCS/MCPQ J 0022 LS SZ 000005 DECK #2 = ?COPY AND COMPARE =/=

CHANGE

The CHANGE system command causes the operating system to change the name of <file-identifier> of specified disk files to another name. If the file name reference resides on a user disk, the disk identifier must be included in <file-identifier>.

If the CHANGE system command is entered and the MCP cannot locate the file, the following message is displayed.

" <file-identifier> " NOT CHANGED - NOT ON DISK

If the CHANGE system command is entered and the file is currently opened by another program, the following message is displayed.

" <file-identifier> " NOT CHANGED - IN USE

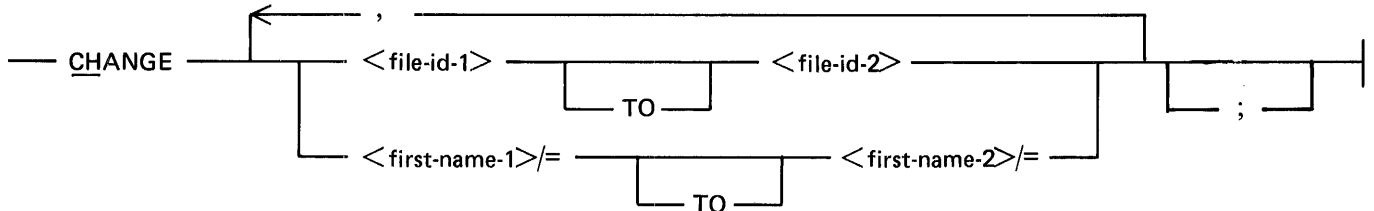
The CHANGE system command is invalid for multipack files.

If <file-identifier> is a PRIVATE file, the CHANGE system command must be prefixed with the USER system command with the proper usercode/password combination.

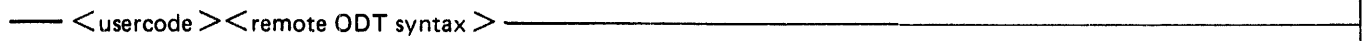
NOTE

If the WFL system option is set, the following syntax is not valid for an unabbreviated CHANGE system command. The WFL syntax is used instead. Refer to the Work Flow Language (WFL) manual, for a description of the WFL syntax.

Remote ODT syntax:



Workstation syntax:



Semantics:

first-name-1

This field, which must contain a valid first name, specifies that the current first name is to be changed. <first-name-1> is changed to <first-name-2> if there are no files on disk with <first-name-2> and none of the files with <first-name-1> are system files or are in use.

first-name-2

This field, which must contain a valid first name, specifies the new first name. <first-name-1> is changed to <first-name-2> if there are no files on disk with <first-name-2> and none of the files with <first-name-1> are system files or are in use.

file-id-1

This field, which must contain a valid file identifier, specifies the name of the file to be changed.

file-id-2

This field, which must contain a valid file identifier, specifies the new name for the file.

Examples:

Command	Response
CHANGE A/B TO C/D;	"A/B" CHANGED TO "C/D"
CH A/B C/D, X Y;	"A/B" CHANGED TO "C/D" "X" CHANGED TO "Y"
CH (FINANCE)/CHECKREG TO (FINANCE)/CHECKS;	"(FINANCE)/CHECKREG" NOT CHANGED - SECURITY ERROR
USER FINANCE/VP CH CHECKREG TO CHECKS;	"CHECKREG" CHANGED TO "CHECKS"
CH MASTER/= OLD_MASTER/=	"MASTER/= " CHANGED TO "OLD_MASTER/= (4 FILES)
CH WORK/WORK_FILE/= TO WORK/OLD_WORK/=	"WORK/WORK_FILE/= " CHANGED TO "WORK/OLD_WORK/= " (6 FILES)

CL (Clear Unit)

The CL system command clears a unit on the system because of an apparent system software loop or hardware malfunction. Any program using a unit cleared by the CL system command is discontinued (DS-ed).

The CL system command can be used to make available a card reader device that has been saved (SV system command) for a particular task.

The CL system command cannot be used with disk devices (for example, DCx, DKx, or DPx).

The CL system command is not to be used to discontinue a program that is accessing the specified device.

Remote ODT syntax:

— CL <unit-mnemonic> —|

Semantics:

unit-mnemonic

This field must contain a valid unit mnemonic on the B 1000 computer system except the unit mnemonics DCx, DKx, DPx, and FDx.

Examples:

Command	Response
CL LPA	LPA CLEARED
CL MTB	MTB CLEARED

CM (Change System Software)

The CM system command designates programs or the (SYSTEM)/USERCODE file as system software with specific functions; for example, as a new MCP or network controller program. This is accomplished by placing the file identifier of the file into an entry in the Name Table on disk. The Name Table is the directory of all operating system software to be used by the clear/start program and other system functions. Each entry in the Name Table implies a specific function for each entered file.

For some files, the actual change resulting from a CM system command is delayed until the following clear/start operation. For example, if a new MCP code file is placed into the M entry of the Name Table, it is loaded during the next clear/start operation. If a new network controller program is designated, it is executed by the MCP whenever the currently-running network controller goes to end of job (EOJ) and a program opens a file with a device type equal to REMOTE.

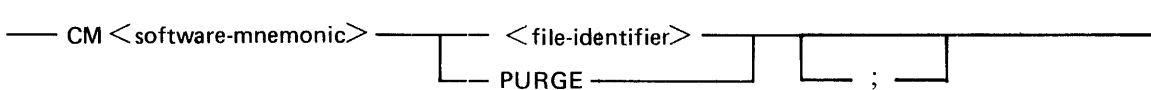
No file having a usercode as a first name can be placed in the Name Table using the CM system command, except in the US slot, which does accept a usercode.

The PURGE option removes the file from the designated Name Table entry.

The SYSTEM/PANDA (PAN) and network controller (C) name table entries cannot be purged. They can only be replaced.

Refer to the clear/start procedure in the B 1000 Systems System Software Operation Guide, Volume 2, for a list of the valid software mnemonics that are used in the Name Table.

Remote ODT syntax:



Semantics:

software-mnemonic

This field can contain any valid software mnemonic listed under the clear/start program in the B 1000 Systems Software Operation Guide, Volume 2. These mnemonics represent a slot in the Name Table for a system program.

file-identifier

This field can contain any valid file identifier and specifies the name of a system program that is to be placed in the Name Table. The PANDA program name table entry must be on system disk.

PURGE

The PURGE keyword is used to remove a file from the designated Name Table slot.

Examples:

Command	Response
CM MX MCP11/OLD;	CLEAR/START REQUIRED TO REPLACE "MCP11" BY "MCP11/OLD" AS MX
CM CX NETCONT/OLD;	EXPERIMENTAL NETWORK CONTROLLER CHANGED FROM "CANDE/HANDLER" TO "NETCONT/OLD"

COPY

The COPY system command executes the SYSTEM/COPY program. Refer to the SYSTEM/COPY program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the COPY system command.

CP (Compute)

The CP system command performs simple arithmetic operations and decimal/hexadecimal conversion on user-input expressions. Parentheses characters can be used to logically group and to nest items within the expression. A special operand "X" can be used to recall the result of the previous CP system command.

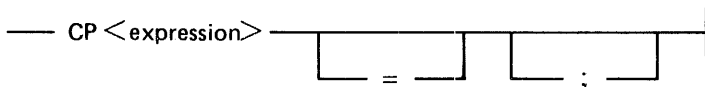
The following are the valid operators recognized by the CP system command.

Operator Mnemonic	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
MOD or M	Modulus (remainder after integer division)

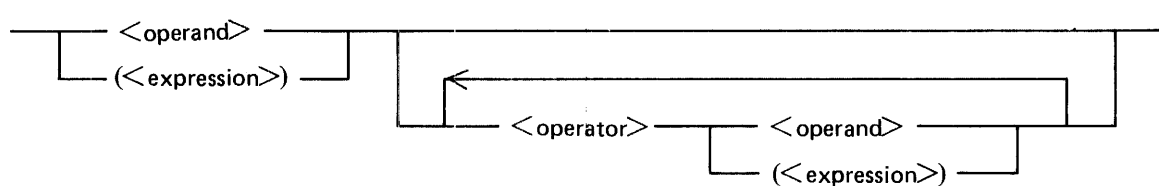
Expressions within parentheses have the highest precedence. The multiplication, division, and modulus operators are at a lower level of precedence. The addition and subtraction operators are at the lowest level of precedence. Operators of equal precedence are evaluated from left to right.

Operands delimited by double quotation mark (") characters are considered EBCDIC data. All other operands and all intermediate results are considered positive integers. Overflow beyond 16,777,215 is truncated. Spaces are required to separate arithmetic operators from the at sign (@) character, which delimits bit strings. Spaces are also required to separate the M operator from any operand.

Syntax:

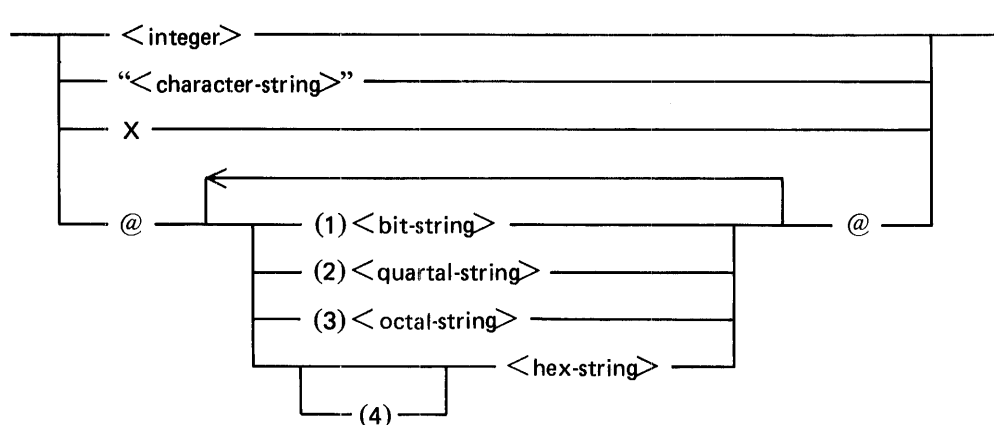


The following is the syntax for <expression>:

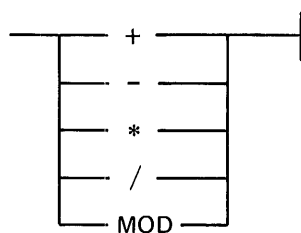


The appearance of <expression> within parentheses indicates that an expression can be used as an operand in a larger expression. This also allows for nesting of parentheses in the expression.

The following is the syntax for an <operand>:



The following is the syntax for an <operator>:



Semantics:

bit-string

This field must contain any bit string of zeros (0) and ones (1).

character-string

This field must contain an EBCDIC character string.

hex-string

This field must contain a hexadecimal number.

integer

This field must contain an integer in the range 0 to 16,777,215 inclusive.

octal-string

This field must contain an octal string.

quartal-string

This field must contain a quartal string.

+, -, *, /, MOD

The +, -, *, /, and MOD keysymbols cause either the addition, subtraction, multiplication, division, or modulus operation to be performed on the operands.

(1), (2), (3), (4)

The (1), (2), (3), and (4) keysymbols specify that a bit, quartal, octal, or hexadecimal string, respectively, immediately follows.

=

The = keysymbol terminates the arithmetic operation.

Examples:

Command	Response
CP @3A@ *4+ @F@	CP: @0000F7@ (247).
CP @F@	CP: @00000F@ (15).
CP X	CP: @00000F@ (15).
CP ((X MOD 8) - 4) * @1A@	CP: @00004E@ (78).
CP @1A@ *4+ @(3)17@ M 16 =	CP: @000077@ (119).
CP 84	CP: @000054@ (84).
CP @F(3)7(1)1111(2)33(4)F0@ +15	CP: @FFFFFF@ (16777215).
CP "*"	CP: @00005C@ (92).

CQ (Clear Queue)

CQ

The CQ system command causes all messages stored in the ODT queue to be cleared, and the ODT screen to be refreshed to its current display state.

Remote ODT syntax:

— CQ — [] ; [] |

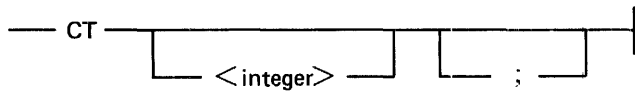
CT (Chip Table)

The CT system command interrogates or changes the size of the table maintained in memory by the MCP and GISMO for logging memory errors. This table exists only on systems with error-correcting memories. On systems without error-correcting memories, the CT system command has no effect.

Entering the CT system command without specifying <integer> causes the MCP to display the current size of the table, as well as the size to be used at the next clear/start operation.

Any change in the table size takes effect when the next clear/start operation is performed.

Remote ODT syntax:



Semantics:

integer

This field, which must contain any integer in the range 0 to 255 inclusive, specifies the number of entries to allocate to the table. Each entry requires four bytes of memory. If <integer> equals 0, the number of entries defaults to one for every 16K bytes of main memory on the system. If <integer> is greater than 255, the table size is set to 255 entries.

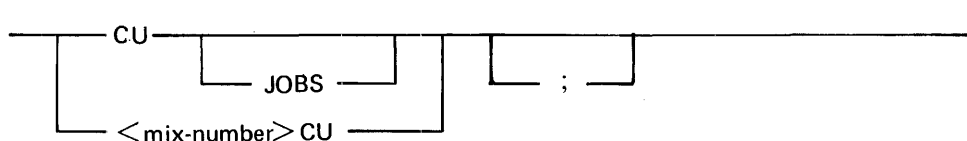
Examples:

Command	Response
CT;	32 SPACES IN CORRECTABLE MEMORY ERROR TABLE WILL BE DEFAULT (1 PER 16K BYTES) SPACES AFTER NEXT CLEAR/START
CT 200;	32 SPACES IN CORRECTABLE MEMORY ERROR TABLE WILL BE 200 SPACES AFTER NEXT CLEAR/START
CT 0;	32 SPACES IN CORRECTABLE MEMORY ERROR TABLE WILL BE DEFAULT (1 PER 16K BYTES) SPACES AFTER NEXT CLEAR/START

CU (Core Usage)

The CU system command displays the amounts of save and overlayable memory in use by a program or displays system totals and the core usage for all programs in the mix. If a <mix-number> is specified, the amount of save and overlayable memory (in bytes) in use by a program is displayed. If a <mix-number> is not specified, the system totals and core usage for all programs in the mix are displayed. The JOBS keyword causes information about each running task to be included in the display, as if each <mix-number> were requested. If the DEBUG MCP option is set, an analysis of linked memory, certain run structure fields, total usercode space, and size and location of user files are also displayed.

Syntax:



Semantics:

mix-number

This field must contain a mix number of a currently executing program. If <mix-number> is not specified, the system totals and the core usage for all programs in the mix are displayed.

JOBS

The JOBS keyword causes information about each running task to be included in the display, as if each <mix-number> were requested.

NOTE

Using the JOBS keyword in a CU system command when the DEBUG MCP option is set results in very lengthy output because a memory analysis of each running task is displayed.

Examples with the DEBUG MCP option reset:

Command	Response
CU	MEMORY USAGE AT 08:18:06.1 SAVE = 167708 BYTES LARGEST OVERLAYABLE = 616513 BYTES AVAILABLE MEMORY: 204923 BYTES; LARGEST AVAILABLE CHUNK: 53627 BYTES. END CU
CU JOBS	MEMORY USAGE AT 08:18:13.5 SAVE = 167708 BYTES LARGEST OVERLAYABLE = 616513 BYTES AVAILABLE MEMORY: 204823 BYTES; LARGEST AVAILABLE CHUNK: 53627 BYTES.

Command	Response
	SAVE = 11535 BYTES; OVERLAYABLE = 15254 BYTES. SAVE = 54623 BYTES; OVERLAYABLE = 61424 BYTES. SAVE = 1841 BYTES; OVERLAYABLE = 54745 BYTES (ROLLED OUT). SAVE = 39157 BYTES; OVERLAYABLE = 72842 BYTES. SAVE = 1403 BYTES; OVERLAYABLE = 80604 BYTES (ROLLED OUT). END CU
234CU	<task-name> = <mix #> SAVE = 1841 BYTES; OVERLAYABLE = 54745 BYTES (ROLLED OUT). END CU

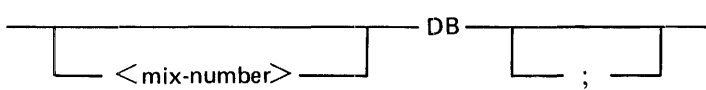
Examples with the DEBUG MCP option set:

Command	Response
CU	MEMORY USAGE AT 08:15:54.4 SAVE = 165060 BYTES LARGEST OVERLAYABLE = 616513 BYTES WITHIN LINKED MEMORY: IN USE = 811397 BYTES OVERLAYABLE = 883515 BYTES AVAIL = 208495 BYTES LARGEST AVAIL = 53627 BYTES SAVE SPACE = 136377 BYTES LINKED MEMORY = 1019892 BYTES FIRST.LINK = @026ECA@ LAST.LINK = @7EEE70@ FENCE = @059539@ NUMBER OF LINKS = 583
234CU	# = 234 , PP = 13, MP = 13 ROLLED OUT RS.Q.IDENT = WATE.Q; RS_NEXT_Q = READY.Q RS.STATUS = 20 ENVIRONMENT #0 (PRIMARY)(CURRENT) P = 0, S = 11, D = 5208 LOCAL DATA SIZE = 36321 BYTES (ROLLED OUT) NUCLEUS ADDRESS = @74BE1E@, DISPLACED = 81677 BYTES CODE OVERLAY COUNT = 86 DATA OVERLAY COUNT = 2 CODE SPACE = 54784 BYTES FILES = 16 0 AT @6F1EC6@, SIZE = 317 BYTES. 1 AT @74FD04@, SIZE = 151 BYTES. 4 AT @6F0D87@, SIZE = 114 BYTES. 5 AT @6F1119@, SIZE = 114 BYTES. 6 AT @6F14AB@, SIZE = 114 BYTES. 7 AT @6F0274@, SIZE = 151 BYTES. TOTAL FILE SPACE = 7699 BYTES SAVE = 1841 BYTES; OVERLAYABLE = 54745 BYTES (ROLLED OUT). END CU

DB (Data Base)

The DB system command displays all the active DMSII data bases. If the mix number of a DMSII program is specified, the name of the data base, number of update operations, number of non-update operations, and total run time are displayed.

Syntax:



Semantics:

mix-number

This field must contain a mix number of a DMSII program that is currently executing. If <mix-number> is not specified, a list of all the active DMSII data bases is displayed.

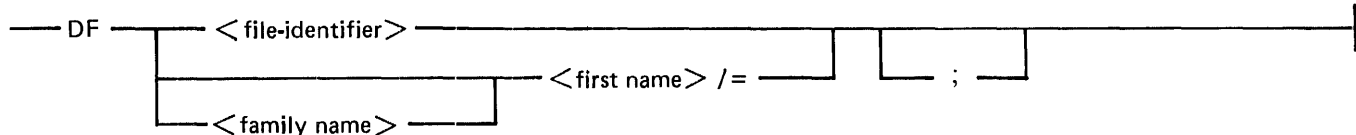
Examples:

Command	Response
DB	NO ACTIVE DATA BASES
DB	ACTIVE DATA BASES ARE B1000TR, PAYROLL
1998DB	(NEWDB) DB/(NEWDB)/TRPROGRAMO = 1998 SZ = 12 DATA BASE B1000TR ACTIVE - 25 UPDATE OPERATIONS AND 87 NON UPDATE OPERATIONS IN 00:00:11.9

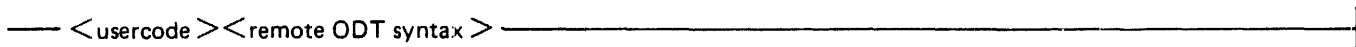
DF (Date of File)

The DF system command displays the compilation date and time for code and interpreter files, and the creation date, time, update date if applicable, and end-of-file (EOF) pointer for all other types of files. The file capacity is included for files of type DATA.

Remote ODT syntax:



Workstation syntax:



Semantics:

family-name

This field, which must contain a valid disk identifier, specifies the name of the user disk to be searched for the files.

first-name

This field, which must contain a valid first name, specifies the first part of a file name.

file-identifier

Refer to section 1 of this guide for a description of file-identifier.

Examples:

Command	Response
DF DMPALL;	DMPALL COMPILED ON 2/25/83 AT 13:39:17.6
DF USER/DATA/FILE;	USER/DATA/FILE CREATED 01/07/81 AT 09:35:06.4, UPDATED ON 04/30/83, EOF = 225, CAPACITY = 300 RECORDS
DF MASTER/=	MASTER/FILE CREATED ON 02/30/83 AT 10:45:07.3 MASTER/LIST CREATED ON 02/30/83 AT 11:52:08.1

DIR

The DIR system command executes the SYSTEM/COPY program. Refer to the SYSTEM/COPY program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the DIR system command.

DL (Disk Location)

The DL system command designates the system default pack for backup files and dump files.

When the DL command is entered with no following phrase, the operating system displays the setting of all the DL types.

When the DL command is entered with a DL type but without a "ON <familyname>" phrase, the setting of that DL type is displayed.

When the DL command is entered with a DL type and a "ON <familyname>" phrase, the setting of that DL type is changed to the specified <familyname>. The <familyname> pack must be online when the DL command is entered. If it is not, a message is displayed and the command is ignored.

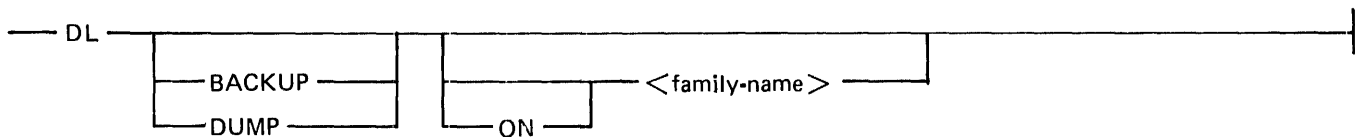
An attempt to power off a pack that is the object of a DL system command results in an error message and the power off command is ignored.

If, at clear/start, a pack that is the object of a DL command is not present, the system displays a message warning the operator that a DL pack is missing.

If a program opens a backup file and the DL BACKUP pack is missing, the program is hung waiting for the missing pack.

If a program dump is requested (<mix#> DP or <mix#> DM) and the DL DUMP pack is missing, a message is displayed and the dumpfile directed to the system disk.

Remote ODT syntax:



Semantics:

BACKUP

The BACKUP keyword refers to the system default pack for printer and punch backup files.

DUMP

The DUMP keyword refers to the system default pack for program dump files. The system dump file always resides on the system disk.

family-name

This field must contain a disk identifier and specify the name of the default disk for the appropriate DL type. A <family-name> of DISK refers to the system disk.

Examples:

Command	Response
DL	DL BACKUP = A DL DUMP = DISK
DL BACKUP	DL BACKUP = A
DL BACKUP ON X	DL BACKUP CHANGED FROM A TO X
DL DUMP ON DISK	DL DUMP ALREADY DISK

DM (Dump Memory and Continue)

The DM system command dumps the contents of the memory space of a program to disk for subsequent analysis by the SYSTEM/IDA program. Processing of the program automatically continues after the memory space of the program is written to disk.

The DM system command creates a file called DUMPFIL/ <integer> on the disk specified by the BD system command. <integer> is incremented by 1 each time a DM system command is performed to make each dump file unique.

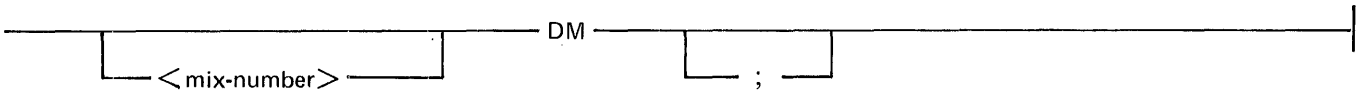
If the program is executed with a usercode/password, the name of the dump file is created with the following format:

<default-family-name>/(<usercode>)/DMP <integer>

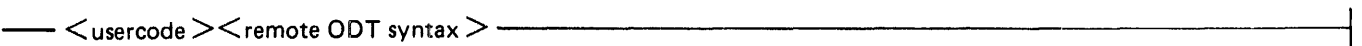
The resulting dump file can be analyzed and printed by the SYSTEM/IDA program. Refer to the PM system command for more information.

If no <mix-number> is specified, and the DBUG system option is set, the memory of the entire system is written to a disk file named SYSTEM/DUMPFIL. A <mix-number> of zero accomplishes the same thing whether or not the DBUG system option is set.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of a program that is currently executing on the system, specifies the program for which the program memory dump is desired. If <mix-number> is not specified, the memory of the entire system is written to a disk file named SYSTEM/DUMPFIL.

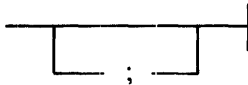
Examples:

Command	Response
DM	SYSTEM MEMORY DUMP COMPLETE
23DM	DMPALL =23 DUMPFIL/10
USER A/B 4531DM	(A) DMPALL =4531 DUMPFIL IS USERPACK/(A)/DMP1014

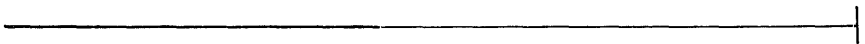
DP (Dump Memory and Discontinue)

The DP system command halts program execution, writes the memory space of the program to disk, and discontinues the program.

Remote ODT syntax:

— <mix-number> DP — 

Workstation syntax:

— <usercode> <remote ODT syntax> — 

Semantics:

mix-number

This field, which must contain a mix number of a program that is currently executing on the system, specifies the program for which the program memory dump and discontinuation is desired.

Example:

Command	Response
267DP;	DMPALL =267 "DUMPFIL/144" DMPALL =267 DS-ED. TIME = 11:27:37.1
USER A/B 4150DP	(A) DMPALL =4150 DUMPFIL IS USERPACK/(A)DMP1019 (A) DMPALL =4150 DS-ED. TIME = 12:01:38.4



DR

DR (Change MCP Date)

The DR system command changes the current date maintained by the MCP. The DR system command functions identically to the DT system command. For inquiry about the current date and time, the TD system command is used. Refer to the TD system command for more information.

The MCP accepts only valid dates. The month entry must be between 1 and 12, the day must be between 1 and 31, and the year must be valid numeric digits. An invalid date or alphabetic input produces an error message.

Remote ODT syntax:

— DR <mm>/<dd>/<yy> _____



Semantics:

<mm>/<dd>/<yy>

This field must contain a valid date, where mm is a 1- or 2-digit month number, dd is a 1-digit or 2-digit day number, and yy is the last two digits of the year.

Example:

Command	Response
DR 7/25/84	DATE = 7/25/84 WEDNESDAY (JULIAN DATE = 84207)

DS (Discontinue Program)

The DS system command discontinues the execution of a program.

The DS system command can be entered at any time after the program reaches beginning of job (BOJ) and prior to end of job (EOJ).

The DS system command stops the execution of the program and returns memory occupied by the program to the system. Any files not previously entered into the disk directory are lost and the disk area occupied is returned to the disk available table. All other files are closed.

Remote ODT syntax:

— <mix-number> DS ————
 └───┬───┘
 └───┬───┘
 ; ———┘

Workstation syntax:

— <usercode><remote ODT syntax> —————┘

Semantics:

mix-number

This field, which must contain a mix number of an executing program, specifies the program to be discontinued.

Example:

Command	Response
271DS	DMPALL =271 DS-ED. TIME = 11:43:47.4

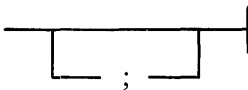
DT

DT (Change MCP Date)

The DT system command changes the current date maintained by the MCP. The DT system command functions identically to the DR system command. For inquiry about the current date and time, the TD system command is used. Refer to the TD system command for more information.

The MCP accepts only valid dates. The month entry must be between 1 and 12, the day must be between 1 and 31, and the year must be valid numeric digits. An invalid date or alphabetic input produces an error message.

Remote ODT syntax:

— DT <mm>/<dd>/<yy> 

Semantics:

<mm>/<dd>/<yy>

This field, which must contain a valid date, where mm is a 1-or 2-digit month number, dd is a 1-digit or 2-digit day number, and yy is the last two digits of the year.

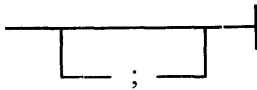
Example:

Command	Response
DT 7/25/84	DATE = 7/25/84 WEDNESDAY (JULIAN DATE = 84207)

ED (Eliminate Pseudo Deck)

The ED system command removes a pseudo-reader file from the pseudo reader and disk directory.

Remote ODT syntax:

— ED <integer> —  —

The diagram shows a horizontal line representing a pseudo-deck. A vertical line extends downwards from the middle of this line, forming a U-shape. Below the horizontal line, there is a semicolon followed by a short horizontal line segment.

Semantics:

integer

This field specifies the pseudo-reader number to be eliminated.

Example:

ED 2;

EM (ELOG Message)

The EM system command writes a message into the ELOG file.

Remote ODT syntax:

EM <message> _____
 └───┬───┘
 └───┘

Semantics:

message

This field contains a message that is to be written in the ELOG file.

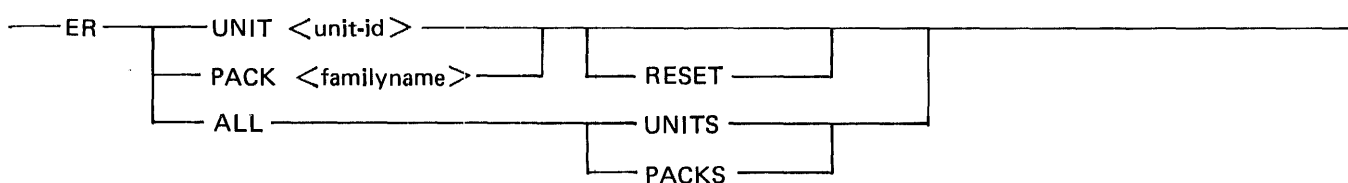
Example:

EM COLUMNS 121 & 122 ARE NOT PRINTING ON LPA

ER (Error Rate)

The ER system command provides error rates for disk units and for removable disk packs. The operator can determine the current rate of I/O errors on a given pack or unit by using the RESET option to zero the counts and then checking the counts over a finite time span after the reset action.

Remote ODT syntax:



Semantics

UNIT

The UNIT <unit> keywords reports the total number of I/O operations and errors recorded for this unit since the last use of the RESET option. (See RESET.) For removable disk drives, this includes all the errors recorded for all the packs that have been mounted on this unit.

PACK

The PACK <familyname> keyword reports the total number of I/O operations and errors recorded for this pack since it was last initialized or since the last use of the RESET option. (See RESET.)

ALL

The ALL keyword reports for all units or packs on the system.

RESET

The RESET keyword resets the I/O operation and error counts for this unit or pack. This option should be used by a Burroughs Field Engineer only.

Examples:

ER UNIT DPA

UNIT DPA:-1000000 OPERATIONS WITH 100 ERRORS SINCE LAST RESET (6/1/84)

UNIT DPA:-10000 OPERATIONS WITH 90 ERRORS SINCE CLEAR/START OR
RESET (10:50.00.1 8/17/84)

ER PACK TWELVE

PACK TWELVE #123456:-100000 OPERATIONS WITH 5 ERRORS SINCE LAST
(6/07/84)

PACK TWELVE #123456:-10000 OPERATIONS WITH 5 ERRORS SINCE POWER
LAST RESET (09:00:00.1 3/18/84)

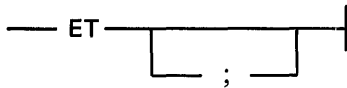
ET

ET (ELOG Transfer)

The ET system command transfers the information in the file SYSTEM/ELOG to the ELOG/#<integer> file. The SYSTEM/ELOGOUT program is automatically executed to analyze and print the ELOG/#<integer> file.

Refer to the SYSTEM/ELOGOUT program in the B 1000 Systems System Software Operation Guide, Volume 2, for more information.

Remote ODT syntax:



FM (Response to Special Forms)

The FM system command is a response to the SPECIAL FORMS REQUIRED message.

The unit mnemonic designates which unit is to be assigned to the file.

The following message is displayed, requiring that an FM system command be entered before the printer or punch file can be opened.

<program-name> = <mix-number> SPECIAL FORMS REQUIRED FOR <file-id>

If the FM system command specifies a backup device, tape or disk, a backup file is created.

Remote ODT syntax:

— <mix-number> FM — <unit-mnemonic> —
 | | | |
 — PACK <family name> — ; —

Semantics:

mix-number

This field must contain a mix number of a program that is suspended and waiting for an FM system command.

unit-mnemonic

This field, which must contain a valid line printer or punch device, specifies the unit on which the special forms are loaded.

PACK <family-name>

This PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

Examples:

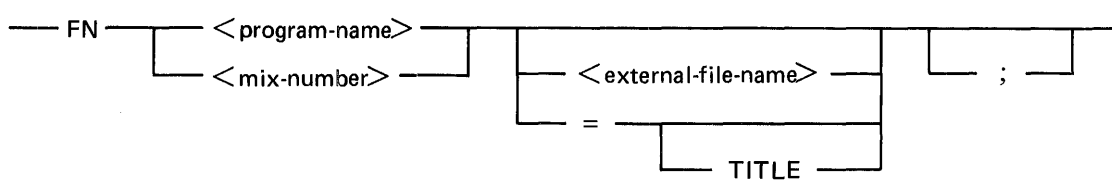
93 FM LPA

94 FM DPB

FN (Display File Name)

The FN system command displays the internal and external file names of an object program. The program can be currently executing, scheduled to be executed, or on disk.

Syntax:



Semantics:

program-name

This field, which must contain a program name of a program that currently exists in the disk directory, specifies the name of the program from which to obtain the internal and external file names.

mix-number

This field, which must contain a mix number of an executing or scheduled to be executed program, specifies the program from which to obtain the internal and external file names.

external-file-name

This field, which must contain a valid external file name of a program, specifies the file for which the internal file name is to be obtained. If this field is not present, then all the file names in the program are displayed.

=

The = key symbol is required when the TITLE keyword is specified and in all other cases is optional.

TITLE

The TITLE keyword causes the external file names to be displayed in title format. A file displayed in title format is in the form B/C ON A, where B is the first part of the file title, C is the second part of the file title, and A is the family name (or disk identifier).

Examples:

Command	Response
FN SYSTEM/MAKEUSER	FILE #0: NEWCODES = NEW/USERCODES FILE #1: LINE = LINE FILE #2: USERCODE = (SYSTEM)/USERCODE FILE #3: PUNCH = NEW/USER.CODES FILE #4: REX_Q = REX_Q
FN SYSTEM/MAKEUSER LINE	FN = "LINE"
FN 4587	FILE #0: LINE = LINE FILE #1: DUMMY = DUMMY FILE #2: USER = USER FILE #3: CARDS = CARDS
FN 123 = TITLE	FILE #0: HELP = TEST/HELP ON USERPACK FILE #1: LINE = LINE

FR (Final Reel of Unlabeled Tape File)

The FR system command notifies the MCP that the last reel of an unlabeled tape file has completed processing, and there are no more input reels to be read.

The FR system command is a response to the following MCP message.

<program-name> <mix-number> FILE <internal-name> REEL #<integer> NOT PRESENT

This message is the result of an unlabeled tape file reaching the end of the reel and the FR system command notifying the program that the file has reached the end of the file.

The FR system command is also valid for labeled tape files, to signal the end of the file without reading all of the reels of the tape file.

The FR system command must be used with paper tape input files to signal the end of the file after all reels have been processed.

Remote ODT syntax:

— <mix-number> FR —————
 | |
 | |
 | |
 | |
 | |

Workstation syntax:

— <usercode><remote ODT syntax> —————

Semantics:

mix-number

This field, which must contain a mix number of a program that is suspended and waiting for another reel of a multireel tape file, specifies a program that has read the last reel of the multireel tape file.

Examples:

MCP Message	Command
DMPALL = 14 FILE INP.FILE (UNLABELED) REEL #5 NOT PRESENT	14FR;
TAPE/PRINT = 33 FILE TAPE (LABELED "TAPFIL") REEL #2 NOT PRESENT	33FR;

FS (Force from Waiting Schedule)

The FS system command forces tasks from the waiting schedule into the active schedule.

Specifying the equal sign (=) character forces all tasks into the active schedule.

The HS system command places a task into the waiting schedule.

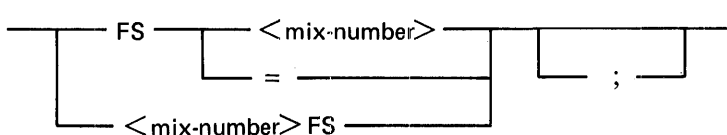
NOTE

The waiting schedule is a schedule of tasks that are waiting to be placed into the active schedule. For example, an EXECUTE program control instruction used with the THEN or AFTER.NUMBER program control instructions places the second program in the waiting schedule.

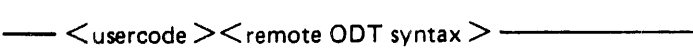
The active schedule consists of those tasks that have satisfied all the requirements for execution and are waiting only for memory space to run.

For a program to be in the mix, it must have reached beginning of job.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of a program currently in the waiting schedule, specifies the program to be placed in the active schedule.

=

The = key symbol causes all tasks in the waiting schedule to be placed into the active schedule.

Examples:

Command	Response
FS 239;	239 FS-ED
240FS;	240 FS-ED
FS =;	250 FS-ED 251 FS-ED

GO (Resume Stopped Program)

The GO system command resumes program execution of a program that has been stopped by the ST system command.

Remote ODT syntax:

— <mix-number> GO ————
 └──┬──┘
 └──┬──┘
 ; —┬──┘

Workstation syntax:

— <usercode><remote ODT syntax> —————┘

Semantics:

mix-number

This field must contain a mix number of a program that is currently suspended by the ST system command.

Example:

Command	Response
123GO	DMPALL = 123 RESUMED TIME = 12:01:34.7

HALT

HALT

The HALT system command causes a B 1000 system to come to an orderly, push-through halt. Upon receipt of the HALT system command the MCP updates the unit and pack IO and error tables for all online disk drives, makes an entry in the ELOG, and makes a request to GISMO to halt the system. GISMO waits until in process input and output operations are complete and then halts the system. Following the halt, the L register contains @000010@. Entering GO on the B 1990 console, or pressing the start button on non-B 1990 systems, causes the system to continue.

System ODT syntax:

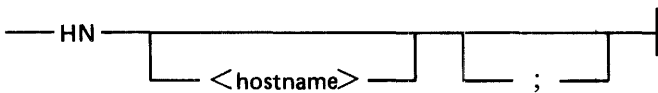
— HALT —|

HN (Hostname)

The HN system command modifies or queries the logical system hostname of the system. The hostname can be modified only when no tasks are running. If there are tasks running when an attempt is made to modify the hostname, the new value of hostname is saved and used at the next clear/start operation or whenever there are no tasks running.

Specifying the HN system command without <hostname> causes the current hostname to be displayed.

Remote ODT syntax:



Semantics:

hostname

This field, which must contain a name of up to 17 characters, specifies the name of the local BNA host.

Examples:

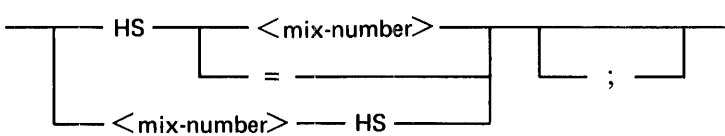
Command	Response
HN B1905;	NEXT NULL MIX OR CLEAR/START WILL USE HOSTNAME = "B1905"
HN;	HOSTNAME = "HOSTTEST", NEXT HOSTNAME "B1905"

HS (Hold in Waiting Schedule)

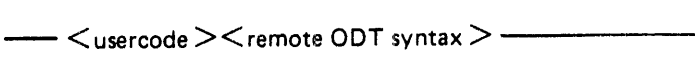
The HS system command places a HOLD on a specific task or tasks, thereby temporarily removing them from the active schedule and placing them in the waiting schedule.

A task that has been placed in the waiting schedule by an HS system command remains in the waiting schedule until an FS system command places the task or tasks in the active schedule.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of a program in the active schedule, specifies the program to be placed in the waiting schedule.

=

The = key symbol removes all tasks in the active schedule and places them in the waiting schedule.

Examples:

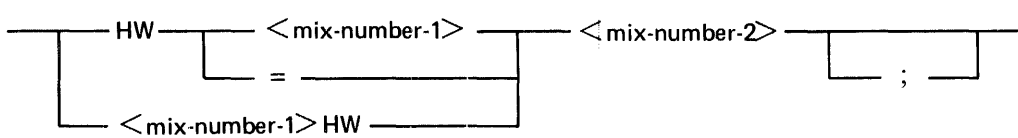
Command	Response
HS 112;	112 HS-ED
115 HS;	115 HS-ED
HS =;	120 HS-ED 121 HS-ED

HW (Hold in Waiting Schedule until Job EOJ)

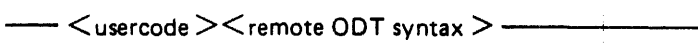
The HW system command places tasks in the waiting schedule to await the end of another task.

A task that has been placed in the waiting schedule by an HW system command remains in the waiting schedule until the program with mix number equal to <mix-number-2> goes to end of task or until the FS system command is performed for the task.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number-1

This field, which must contain a mix number of a program in the active schedule, specifies the program that is to be placed in the waiting schedule.

mix-number-2

This field, which must contain a mix number of a program in the mix, active schedule, or waiting schedule, specifies the task that must go to end of job before the tasks in the waiting schedule can be placed in the active schedule.

=

The = key symbol causes all tasks in the active schedule to be placed into the waiting schedule until the program with mix number equal to <mix-number-2> goes to end of job.

Examples:

Command	Response
HW 190 183	190 HW-ED
200 HW 201	200 HW-ED
HW = 400	390 HW-ED
	391 HW-ED

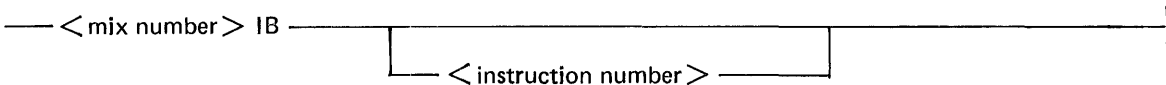
IB (Instruction Block)

The IB system command displays a requested instruction block for a WFL job.

The instruction block is displayed in the following format:

INSTRUCTION <instruction number>: text of instruction

Syntax:



The <instruction number> must be an integer constant in the range of 1 to 63 inclusive.

If no <instruction number> is given with the IB system command, the most recently executed <INSTRUCTION statement> is displayed. If an <INSTRUCTION statement> has not been executed, a message is displayed indicating that fact.

Examples:

2657IB 1

JOB2657 =2657 INSTRUCTION 1: TESTTAPE IS IN TAPE RACK 3.

3945IB

IB/JOB =3945 INSTRUCTION 7: IF T17 OR T17A WERE NOT COPIED
FROM TESTTAPE TO USERS, PLEASE DS THIS JOB
AND LEAVE JK A NOTE.

IC (Interpreter Count)

The IC system command interrogates or changes the number of entries allocated to the MCP Interpreter Dictionary.

The Interpreter Dictionary is a table maintained by the MCP that contains information concerning all active interpreters. One entry is required in this table for each interpreter running on the system, plus one entry each for the MICRO-MCP and GISMO interpreters. In addition, because the MCP is always active, one entry is reserved for the SDL2 interpreter. Each entry in the Interpreter Dictionary requires 28 bytes of non-overlayable memory space.

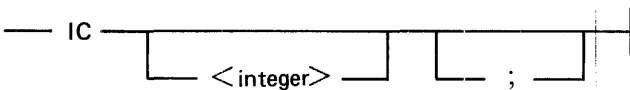
The Interpreter Dictionary is constructed and initialized during the clear/start operation by the SYSTEM/INIT program, and contains the number of entries specified by <integer>. The default value following a coldstart operation is 7, implying that the maximum number of interpreters that can be active on the system at one time is four (because an entry is reserved for the MICRO-MCP, GISMO, and SDL2 interpreters).

The IC system command allows the Interpreter Dictionary size to be changed if it becomes necessary to increase or decrease the maximum number of active interpreters allowed.

No tasks can be in the mix or in either the waiting or active schedules when changing the interpreter count, and a clear/start operation is required immediately after the IC system command is entered.

If the IC system command is entered without specifying <integer>, the MCP displays the current value of the interpreter count.

Remote ODT syntax:



Semantics:

integer

This field, which must contain an integer in the range 3 to 31 inclusive, specifies the number of entries to be allocated to the Interpreter Dictionary by the SYSTEM/INIT program during the next clear/start operation.

Examples:

Command	Response
IC;	INTERPRETER COUNT = 7
IC 4;	CLEAR/START REQUIRED TO CHANGE INTERPRETER COUNT FROM 7 TO 4

IL (Ignore Label)

The IL system command causes the label to be ignored on the file mounted on the designated unit.

The IL system command can be used in response to the following MCP messages:

NO FILE <file-identifier>

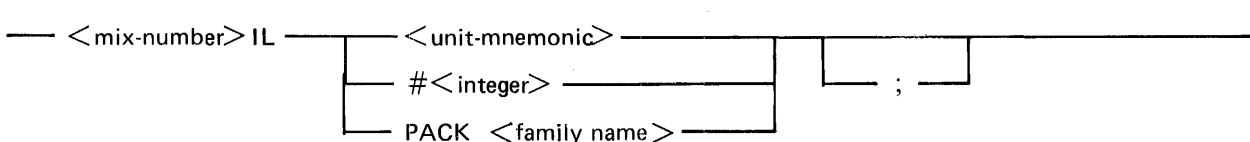
DUPLICATE INPUT FILE <file-identifier>

<file-identifier> NOT IN DISK DIRECTORY

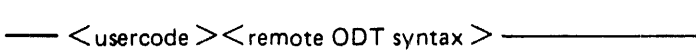
It is assumed that the system operator knows that the file on the unit selected is the correct file regardless of the original location of the file. If <unit-mnemonic> specifies a disk drive, the directory on that drive is searched for the required file identifier. Specifying #<integer> designates a pseudo-reader file (by number) as the input device.

The IL system command referencing a card reader device that has been reserved by mix number for a program, overrides the reservation and assigns the card reader device regardless of the file identifier requested.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of an executing program, specifies the program that is to ignore the label of the file mounted on the designated unit.

unit-mnemonic

This field, which must contain a valid unit mnemonic that is an input unit, specifies the device in which the label is to be ignored.

integer

This field must be an integer of a pseudo-reader file currently on disk, specifies that the pseudo reader is to be used as the input file.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

Example:

MCP Message	Command
DMPALL = 300 USER/MASTER/FILE NOT IN DISK DIRECTORY	300IL DPA;

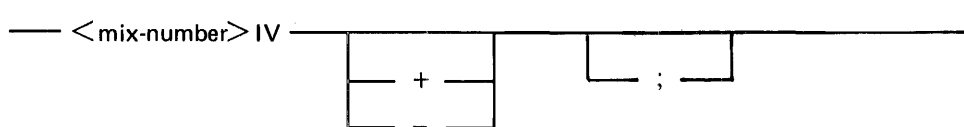
IV (Invisible)

The IV system command makes a task either transparent or visible to the normal MX and WY system commands. The IV system command can be used to reduce ODT traffic by hiding system-oriented programs that remain in the mix but whose status is not usually of interest to the operator.

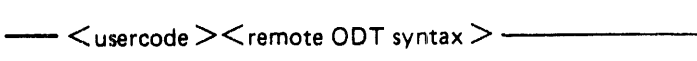
If the optional + or - keysymbols are not specified, the invisibility status is reported. The + keysymbol makes the task invisible and the - keysymbol restores normal visibility. Invisible tasks are reported by the MX and WY system commands in the following cases:

1. The mix number of the invisible task prefaces the command, such as 123 WY.
2. The command is prefaced by the USER system command.
3. The ALL keyword follows the command.
4. The task is waiting operator input or action.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of an executing program, specifies the program for which the invisibility status is being changed or inquired.

+

The + keysymbol causes the task specified by <mix-number> to be invisible.

-

The - keysymbol causes the task specified by <mix-number> to be restored to normal visibility.

Examples:

Command	Response
238 IV	CANDE =238 JOB NOW VISIBLE.
238 IV +	CANDE =238 JOB NOW INVISIBLE.
238 IV-	CANDE =238 JOB NOW VISIBLE.

JOBSTART

The JOBSTART system command requests that the task file on disk specified by <file-identifier> be sent to the host in a BNA network specified by <host-name>, and that the task be run there. The BNA system must be running locally for this instruction to be valid. The syntax within the task file must be valid syntax for the receiving host system. The local BNA system does not syntax-check the information in the task file.

Syntax:

— JOBSTART <file-identifier>; HOSTNAME = <host-name> 

Semantics:

host-name

This field must contain a valid BNA host name of a BNA host computer system currently in the BNA network.

file-identifier

This field, which must contain a valid file identifier, specifies the name of the task file. The instructions contained in this file must be valid on the receiving BNA host system.

Examples:

```
USER TEST/TEST JOBSTART REMOTE/DECK;HOSTNAME = B6900;
```

```
USER PROD/USER TJOBSTART TEST/JOB-FILE;HOSTNAME = B1985;
```



JS (Jiggle Schedule)

The JS system command requests the MCP to execute the top entry in the active schedule.

The MCP checks the active schedule as each program goes to end of task or when a program is scheduled and executes programs that are in the active schedule, if possible. The JS system command causes the MCP to check the active schedule when the instruction is entered.

Syntax:

— JS —————
 └───┬───┘
 ; ───┘

Example:

JS;

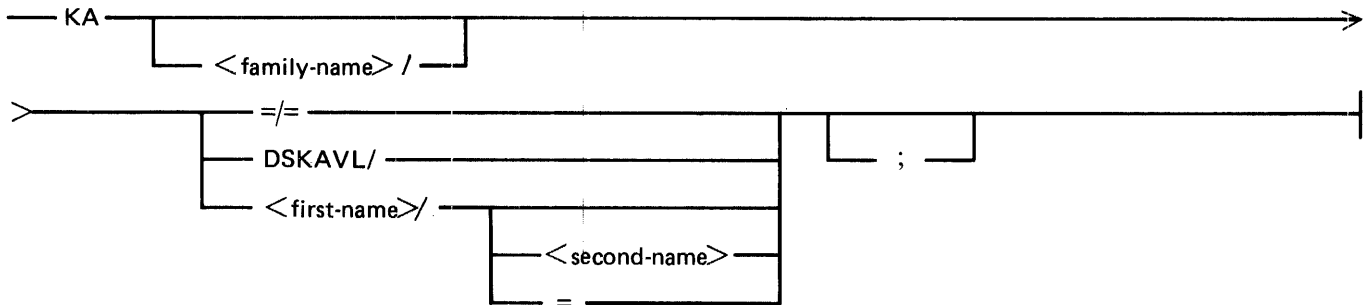
KA (Analyze Disk Directory)

The KA system command allows the system operator to analyze the contents of a disk directory, including file area assignments.

If a disk identifier is included in the KA system command, information is displayed for the specified disk pack or disk cartridge; otherwise, information is displayed for the system disk. If the DSKAVL keyword is included in the command, the available disk areas are listed.

If the SYSTEM/PANDA program is on disk, it is initiated whenever a KA system command is entered with the =/=, <first-name>/= or DSKAVL parameters. If the SYSTEM/PANDA program is not on disk, the MCP analyzes the disk directory and displays the results.

Remote ODT syntax:



Semantics:

family-name

This field, which must contain a valid disk identifier, specifies the user disk in which to analyze the disk directory.

=/=

The =/= keysymbol causes all the files on the SYSTEM disk or the user disk specified by <family-name> to be analyzed.

DSKAVL

The DSKAVL keysymbol causes the available disk areas to be listed.

first-name

This field, which must contain a first name, specifies the main-directory name in the disk directory to be analyzed.

second-name

This field, which must contain a 10-character name, specifies the subdirectory name for the file to be analyzed.

=

The = keysymbol causes all the files having the specified <first-name> to be analyzed.

Examples:

KA DMPALL;

KA USER/DSKAVL/;

KA MASTER/=;

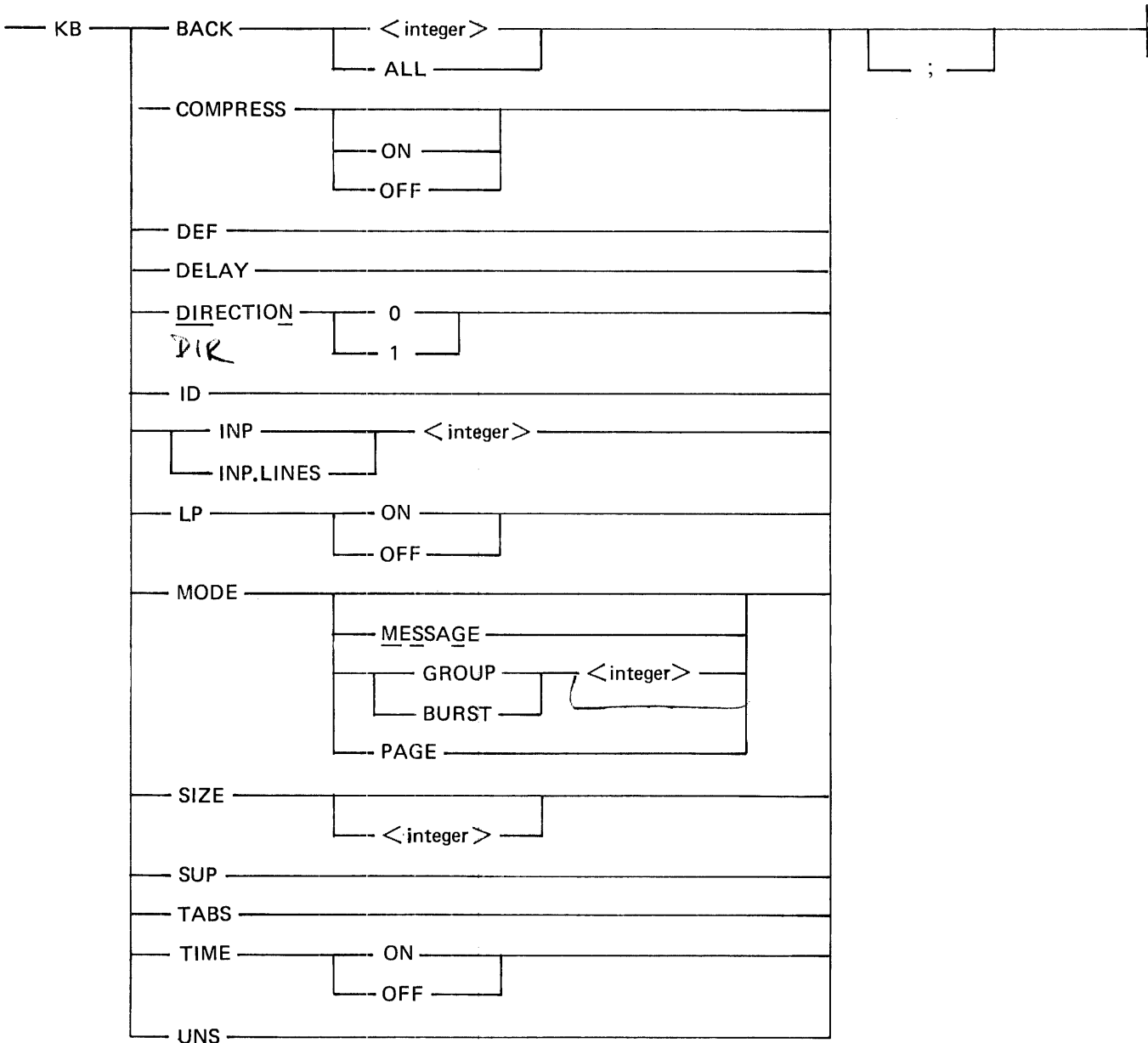
KA DSKAVL;

KA =/=;

KB (Print ODTLOG)

The KB system command specifies certain characteristics for displaying the ODT information.

Remote ODT syntax:



Semantics:

BACK

The BACK keyword causes the last <integer> number of sectors in the 200-sector ODT queue file to be formatted and displayed, a screenful at a time, on an ODT. If the ALL keyword is included, all 200 sectors are displayed. If LP ON has been specified, the same messages are directed to a line printer file by the SYSTEM/ODT program.

KB

COMPRESS

Interrogates, causes (with ON), or stops (with OFF) the replacement of multiple blanks with a single blank in ODT messages.

DEF

The DEF key symbol restores the default ODT settings. The default settings are INP 2, DIRECTION 0, UNS, LP OFF, TIME OFF.

DELAY

The DELAY keyword specifies the tenths of seconds that SYSTEM/ODT waits to display messages from the MCP. This specification is valid in MODE GROUP only. Collected messages are displayed if any of the following occurs:

- a DELAY of <integer> tenths of seconds.
- a GROUP of <integer> messages is collected.
- a full page of messages is collected.

This feature is a subtle fine-tuning tool.

DIRECTION

The DIRECTION keyword specifies the order in which ODT messages are to appear on the screen. If KB DIRECTION 0 is specified, the most recent message appears at the top of the ODT output area, with earlier messages appearing in reverse order downward. If KB DIRECTION 1 is specified, the most recent message appears at the bottom line of the ODT output area, with earlier messages appearing toward the top.

HELP

The HELP keyword provides the operator with a help-screen of all the KB options, and a brief description of each option.

ID

The ID keyword provides the operator with the compilation date and patch level of the SYSTEM/ODT program.

INP

The INP keyword changes the size of the input area reserved at the top of the screen. The default size of the input area is two lines. The number of reserved lines is specified by the value of <integer>, which must be in the range 2 to 10 inclusive.

Changing the size of the input area affects the size of the output area inversely. Increasing the size of the input area decreases the size of the output area proportionately.

The input area remains displayed on the screen and in the memory of the ODT itself until a different message is entered. If the operator is informed of an error in a long system command, the entire command does not have to be retyped. By positioning the cursor at the portion of the system command containing the error and correcting it, the message can be retransmitted in its error-free form.

LP

Specifying KB LP ON causes a line printer file to be opened and all ODT message traffic to be written to the line printer file as well as to the ODT device. Specifying KB LP OFF restores normal ODT-only display. The default is LP OFF. The line printer file is closed when KB LP OFF is entered.

MODE

The MODE keyword is used to either interrogate or change the manner in which messages are displayed on the ODT. The current MODE is interrogated and displayed when the KB MODE command is entered with no additional parameters.

The MESSAGE or MSG mode causes each output message to be scrolled on the ODT screen; an ETX character is optional after each operator system command.

The GROUP or BURST mode causes output messages to be collected but not displayed until one of the following conditions is true: 1) <integer> messages are received, 2) a full screen of messages is received, or 3) one-tenth of a second elapses without a message being received. <integer> has a default value of 15, but can be given a value in the range 2 to 22 inclusive. An ETX character is optional after each operator system command. The GROUP or BURST mode maximizes the performance of most remote and hard ODT configurations.

The PAGE mode causes the entire ODT screen to be refreshed at one time. An ETX character is required after each operator system command. The PAGE mode is intended and mandatory for the older model B-9248-2 ODT devices.

SIZE

The SIZE keyword either interrogates or changes the size of the ODT queue file. The default size of the file set at the time of a coldstart operation is 200 sectors. If <integer> is not specified, the current size of the queue file is interrogated and displayed. The <integer> parameter is used to change the size of the ODT queue file to a value between 100 and 1024 sectors inclusive. Larger queue file sizes are advantageous for installations with high system activity, where wrap-around of the queue file occurs rapidly.

SUP

The SUP key symbol causes all messages on the screen, except system commands entered at the ODT device and output messages, to be suppressed.

TABS

The TABS keyword is used to set a tab stop every ten positions on the ODT. The ODT firmware must permit adjustable tab stops.

TIME

The TIME keyword is used with ODT devices and causes the time at which the system command or output message was entered to be displayed on the ODT. The time portion of the message is displayed in front of the message, and is always included on each message stored in the ODT queue, even if the option is not set. Specifying KB TIME ON sets this option and specifying KB TIME OFF resets this option.

UNS

The UNS key symbol causes all messages from ZIP commands, QUEUE commands, control cards from card readers and pseudoreaders, and so forth, to be displayed on the ODT.

Examples:

KB BACK 20;

KB TIME ON;

KB LP ON;

KB SUP;

KB DIRECTION 1;

KB MODE GROUP 15

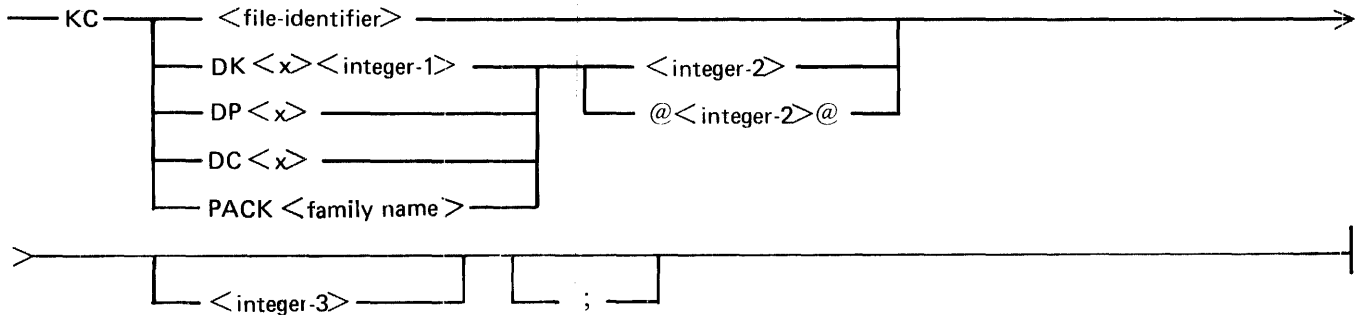
KB TABS

KB SIZE 800

KC (Print Disk Segments in Character Format)

The KC system command prints selected files or segments in a character format on a line printer.

Remote ODT syntax:



Semantics:

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

DC

The DC keysymbol specifies that the disk sectors to be printed reside on a disk cartridge.

DK

The DK keysymbol specifies that the disk sectors to be printed reside on a head-per-track disk.

DP

The DP keysymbol specifies that the disk sectors to be printed reside on a disk pack.

integer-1

This field, which must contain a non-zero integer, specifies the electronics unit of the head-per-track disk.

integer-2

This field, which must contain an integer, specifies the disk address from which printing is to begin. If the at sign (@) character delimits <integer-2>, then the disk address must be expressed as a hexadecimal number.

integer-3

This field, which must contain an integer, specifies the number of disk segments to print. The default value is 1. The maximum value is 100.

x

This field, which must contain an upper-case character and must be concatenated with the DC, DK, or DP keysymbol, specifies the disk unit on which the disk sectors to be printed reside.

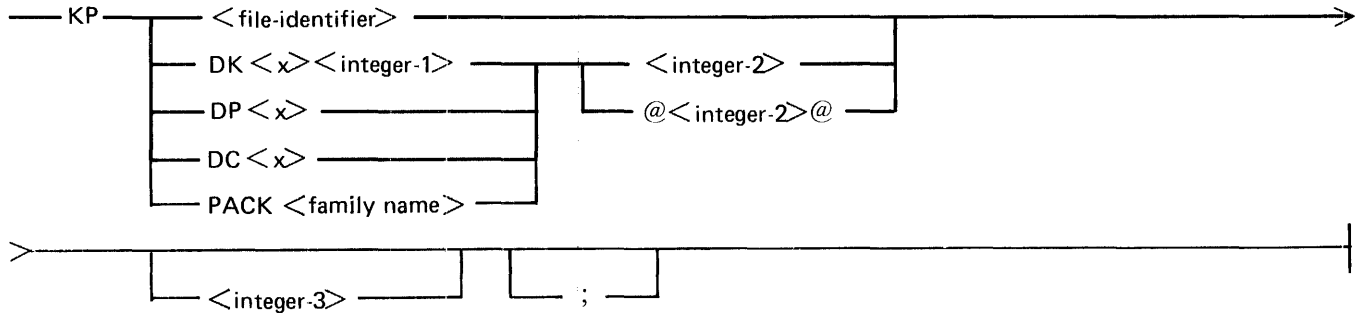
Examples:

Command	Result
KC A/B 10;	Prints 10 segments of file A/B.
KC A/B;	Prints 1 segment of file A/B.
KC CCC/X/;	Prints 1 segment of file X on user disk CCC.
KC DPA @5C@ 15;	Prints 15 segments from the hexadecimal disk address 5C on disk DPA.
KC DKA 1 200 10;	Prints 10 segments on EU 1 from the decimal disk address 200.

KP (Print Disk Segments in Hexadecimal Format)

The KP system command prints selected files or segments in a hexadecimal format on a line printer.

Remote ODT syntax:



Semantics:

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

DC

The DC keysymbol specifies that the disk sectors to be printed reside on a disk cartridge.

DK

The DK keysymbol specifies that the disk sectors to be printed reside on a head-per-track disk.

DP

The DP keysymbol specifies that the disk sectors to be printed reside on a disk pack.

integer-1

This field, which must contain a non-zero integer, specifies the electronics unit of the head-per-track disk.

integer-2

This field, which must contain an integer, specifies the disk address from which printing is to begin. If the at sign (@) character delimits <integer-2>, then the disk address must be expressed as a hexadecimal number.

integer-3

This field, which must contain an integer, specifies the number of disk segments to print. The default value is 1. The maximum value is 65.

x

This field, which must contain an upper-case letter and must be concatenated with the DC, DK, or DP keysymbol, specifies the disk unit on which the disk sectors to be printed reside.

Examples:

Command	Result
KP A/B 10;	Prints 10 segments of file A/B.
KP A/B;	Prints 1 segment of file A/B.
KP CCC/X/;	Prints 1 segment of file X on user disk CCC.
KP DPA @5C@ 15;	Prints 15 segments from the hexadecimal disk address 5C on disk DPA.
KP DKA 1 200 10;	Prints 10 segments on EU 1 from the decimal disk address 200.

LC (Log Comment)

The LC system command places a message into the system log file.

The message starts in the first position after the LC system command and continues to the end-of-text (ETX) character.

Remote ODT syntax:

— LC <log-message> _____
 └───┬───┘
 ;

Semantics:

log-message

This field must contain a message that is desired in the system log file.

Example:

LC OPERATOR JOHN JONES ON AT 8:00 AM;

LD (Pseudo Load)

The LD system command initiates the building of pseudo-reader files on disk to be processed by pseudo readers.

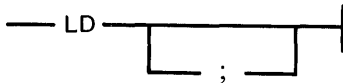
The LD system command causes the SYSTEM/LDCONTRL program to be executed. The SYSTEM/LDCONTRL program requires a ?DATA CTLDCK program control instruction, followed by the card deck, and is terminated with a ?ENDCTL program control instruction.

The file identifier of the card deck is assigned by a ?DATA <file-identifier> program control instruction preceding the data deck to be read. Each data deck that is loaded is numbered consecutively along with its file identifier, which is used in opening the pseudo-reader files.

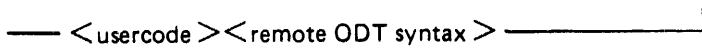
Users can create pseudo-reader files out of disk data files by file-equating the internal file name of the SYSTEM/LDCONTRL program, CARD, to their respective disk files (for example, LD; FILE CARD NAME USER/MASTER/FILE DISK DEFAULT;).

Statements in the pseudo-reader file must be terminated by either a semicolon (;) character or a percent sign (%) character if the file is a sequenced disk file.

Remote ODT syntax:



Workstation syntax:



Example 1:

```

CONTROL DECK
    ? DATA CTLDCK
DECK A [ ? COMPILER <program-name> COBOL74 SYNTAX
        ? DATA CARDS
        •
        • (data deck)
        •
        ? END
DECK B [ ? COMPILER <program-name> FORTRAN77 LIBRARY
        ? DATA CARDS
        •
        • (data deck)
        •
        ? END
DECK C [ ? DATA file-id
        •
        • (data deck)
        •
        ? END
        ? ENDCTL
    
```

Example 2:

```

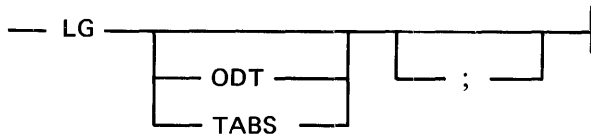
DECK A [ ? EXECUTE PROG10; SWITCH 0 = 1;      %
        ? FILE DISK NAME DATAFILE;        %
        ? FILE TAPE1 NAME DIFFNAME;        %
        ? END                                %
DECK B [ ? EXECUTE PROG20; AFTER PROG10; %
        ? FILE T NAME DIFFNAME;
        ? FILE LINE NO HAR REP 2;
        ? END                                %
        ? ENDCTL
    
```

LG (Transfer and Print Log)

The LG system command transfers and prints either the system log or the ODT log file. The LG system command functions identically to the LN system command. The log files are numbered sequentially starting with LOG/#000001 or ODTLOG/<date-and-time>, where <date-and-time> has the form <MMDDYYHHMM>. The program SYSTEM/LOGOUT, SYSTEM/ODTLOGOUT, or TABS/LOGOUT is executed, performing the necessary file equation to print the specified log file. The program must be in the disk directory in order for the MCP to process the LG system command.

If the ODT or TABS keywords are not included in the LG system command, the system log file is transferred and the SYSTEM/LOGOUT program is executed.

Remote ODT syntax:



Semantics:

ODT

The ODT keyword causes the ODT log file to be transferred and the SYSTEM/ODTLOGOUT program to be executed.

TABS

The TABS keyword causes the system log file to be transferred and the TABS/LOGOUT program to be executed.

Examples:

LG;

LG TABS;

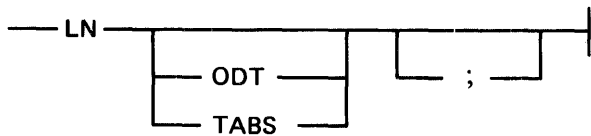
LG ODT;

LN (Transfer and Print Log)

The LN system command transfers and prints either the system log or the ODT log file. The LN system command functions identically to the LG system command. The log files are numbered sequentially starting with LOG/#000001 or ODTLOG/<date-and-time>, where <date-and-time> has the form <MMDDYYHHMM>. The program SYSTEM/LOGOUT, SYSTEM/ODTLOGOUT, or TABS/LOGOUT is executed, performing the necessary file equation to print the specified log file. The program must be in the disk directory in order for the MCP to process the LN system command.

If the ODT or TABS keywords are not included in the LN system command, the system log file is transferred and the SYSTEM/LOGOUT program is executed.

Remote ODT syntax:



Semantics:

ODT

The ODT keyword causes the ODT log file to be transferred and the SYSTEM/ODTLOGOUT program to be executed.

TABS

The TABS keyword causes the system log file to be transferred and the TABS/LOGOUT program to be executed.

Examples:

LN

LN TABS;

LN ODT;

LP (Lock Protection)

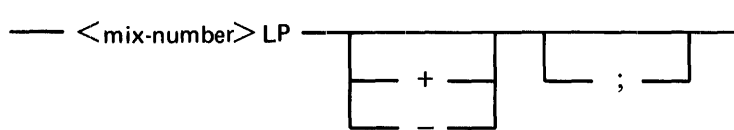
The LP system command interrogates or changes the PROTECTION program attribute for a running program. Setting the PROTECTION program attribute helps to prevent a running program from being accidentally tampered with by the following system commands.

CL
DP
DS
QC
ST
SW

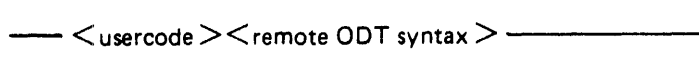
A program can have the PROTECTION program attribute set at execution time by specifying the PROTECTION file attribute in the EXECUTE program control instruction.

Specifying the LP system command without the plus sign (+) or minus sign (-) character interrogates the PROTECTION file attribute and displays the current setting.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field must contain a mix number of a task currently in the mix.

+

The + keysymbol causes the PROTECTION program attribute to be set.

-

The - keysymbol causes the PROTECTION program attribute to be reset.

Examples:

Command	Response
334LP;	SYSTEM/ODTLOGOUT =334 LOCK FLAG OFF
334LP+;	SYSTEM/ODTLOGOUT =334 LOCK FLAG ON
334LP-;	SYSTEM/ODTLOGOUT =334 LOCK FLAG OFF

LO

LT (Load Translator)

The LT system command changes the translate table for the 400 and 750 line-per-minute (LPM) Train Printers. Train printers have interchangeable print train modules that allow a variety of specialized character sets to be used. Each print module requires a unique translator to be loaded into the printer control.

The 1100 and 1500 LPM Train Printers require the B 1247-4 Printer Control and have an automatic train identification feature that allows the MCP to immediately recognize the print translator required, and to load it into the printer control when necessary.

The 400 and 750 LPM Train Printers require either the B 1247 or B 1247-4 Printer Controls and do not have the automatic feature. These train printers require notification from the system operator whenever the train module is changed. This is accomplished by the LT system command.

The characteristics of the printer translate tables and the LT system command depend on which printer control (B 1247 or B 1247-4) is being used. The printer control identification can be determined from a SYSTEM/ELOGOUT printer listing.

B 1247 Train Printer Control (Control Identification = @10@)

If the B 1247 Printer Control (not allowed with the 1100 and 1500 LPM Train Printers) is used, the file SYSTEM/PRINTCHAIN is created automatically by the MCP on the SYSTEM disk. The MCP loads a translator from this file into the printer control based on the setting of the TRAIN SELECTOR SWITCH on the printer. The following table shows the possible switch settings and their associated translators.

Train Selector Switch Setting	Translator
1	64-character EBCDIC
2	48-character EBCDIC
3	16-character EBCDIC
4	96-character EBCDIC
5	48-character FORTRAN
6	48-character B300/B500
7	48-character RPG
8	48-character FORTRAN-NONSTD

The MCP loads the translator specified by the switch setting the first time the printer goes ready following a clear/start operation. If it is necessary to change the character set, mount the new train module in the printer, select the proper switch setting, make the printer ready, and enter the LT system command to notify the MCP of the presence of the new character set. Specifying <train-identification> in the LT system command is not valid, because the MCP uses the setting of the TRAIN SELECTOR SWITCH to determine the proper translator.

LT

B 1247-4 Train Printer Control (Control Identification = @3E@)

If the B 1247-4 Printer Control is used, the file SYSTEM/TRAINABLE must be present on the SYSTEM disk. This file is created by the SYSTEM/BUILDTRAIN program, and must contain all print translators required by an installation. Refer to the SYSTEM/BUILDTRAIN program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the SYSTEM/BUILDTRAIN program and a list of the standard print translators and their train identifications.

For 400 and 750 LPM Train Printers connected to the B 1247-4 Printer Control, the TRAIN SELECTOR SWITCH is ignored, and the MCP loads the translator specified by the LT system command into the printer control.

The MCP displays the following message the first time a 400 or 750 LPM Train Printer connected to a B 1247-4 Printer Control goes ready after a clear/start operation.

"LT" REQUIRED FOR <line-printer-unit-mnemonic>

A translator is loaded by the MCP only when requested to do so through the LT system command. Once loaded, the MCP does not change the translator until another LT system command is entered.

The first time that an 1100 or 1500 LPM Train Printer goes ready after a clear/start operation, the MCP automatically loads the translator specified by the train module identification. To change a train module, it necessary to mount the new module in the printer, make the printer ready, and enter the RY system command to notify the MCP of the change.

If <train-identification> is omitted from the LT system command, the MCP displays the train identification number of the train currently being used.

NOTE

The interrogation feature is allowed only with the B 1247-4 Printer Control.

Remote ODT syntax:

— LT <unit-mnemonic> ————— |
<train-identification>	

Semantics:

unit-mnemonic

This field, which must contain a valid line-printer unit mnemonic, specifies to the operating system the line printer that has the new translator.

train-identification

This field must contain a valid train identification. Refer to the SYSTEM/BUILDTRAIN program in the B 1000 Systems System Operation Guide, Volume 2, for a list of the valid train identifications.

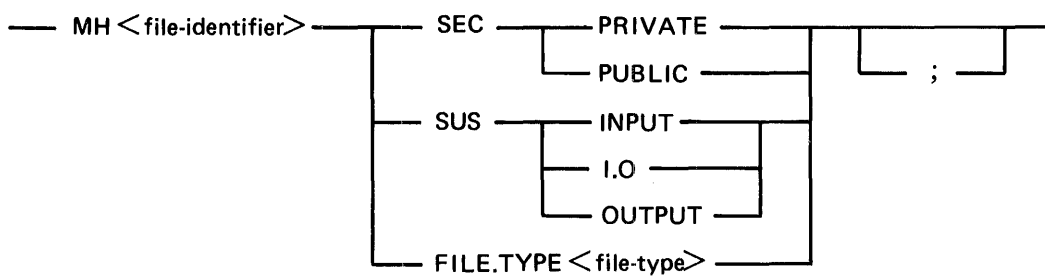
Examples:

Command	Response
LT LPA;	"005" WAS SELECTED ON LPA
LT LPB 255;	EBCDIC3.64 = "255" LOADED ON LPB
LT LPC EBCDIC96;	EBCDIC96 = "016" LOADED ON LPC

MH (Modify Header)

The MH system command changes the security attributes or file type of a disk file. If access to a file is currently restricted in any way (PRIVATE, INPUT, or OUTPUT), the MH system command must be preceded by a USER system command with the proper usercode/password pair or a PRIVILEGED usercode/password pair.

Remote ODT syntax:



Workstation syntax:



Semantics:

SEC

The SEC keysymbol specifies the access rights to the file. The access rights can be set to PUBLIC or PRIVATE. The PRIVATE keyword specifies the access rights to the file to be a proper usercode/password pair or a PRIVILEGED usercode/password pair. The PUBLIC keyword specifies the access rights to the file to be from any program.

SUS

The SUS keysymbol limits the type of access to a file. The type of access can be INPUT, I.O (input and output), or OUTPUT. The INPUT keyword limits the type of access for a file to input only. If an object code file is marked as INPUT, it is considered executable and cannot be opened as a data file by a program. The I.O keysymbol allows the file to be opened input or output by another program. The OUTPUT keyword limits the type of access for a file to output only.

FILE.TYPE

The FILE.TYPE keysymbol allows the file type of a disk file to be changed to <file-type>, where <file-type> can have any of the following values.

BASIC	FORTRAN77	RPG
COBOL	IBASIC	SDL
COBOL74	JOBS	SDL2
DASDL	MIL	SEQD
DATA	NDL	SORT
FORTRAN	PASCAL	

Examples:

Command: MH MASTER/FILE SEC PRIVATE;
Response: "MASTER/FILE SECURITYTYPE SET TO PRIVATE

Command: USER PRIVIL/USER MH (PRIVATE)/FILE SUS INPUT SEC PUBLIC;
Response: "(PRIVATE)/FILE" SECURITYUSE SET TO INPUT "(PRIVATE)/FILE SECURITYTYPE SET TO PUBLIC

Command: US USER/A MH TESTFILE SUS I.O
Response: "TESTFILE" SECURITYUSE SET TO I.O

Command: MH TEST/FILE FTP DATA
Response: TEST/FILE FILETYPE CHANGED TO DATA

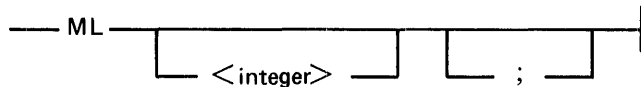
ML (Mix Limit)

The ML system command interrogates or changes the maximum number of low-priority tasks allowed concurrently in the mix. Any further attempt to execute low-priority programs after the mix limit has been reached causes those programs to remain in the active schedule. The only tasks that are allowed to start after the mix limit has been reached are those having a PROCESSOR.PRIORITY program attribute of 9 or greater (crashout priority) and those having a SCHEDULE.PRIORITY program attribute of 15 (tasks).

If <integer> is omitted, the current setting of the mix limit is interrogated and displayed.

The coldstart programs (COLDSTART/DISK and COLDSTART/TAPE) set the mix limit to 63.

Remote ODT syntax:



Semantics:

integer

This field, which can contain any integer in the range 0 to 255 inclusive, specifies the maximum number of tasks allowed in the mix at a priority of eight or less. Specifying the ML system command without <integer> displays the current mix limit.

Examples:

Command	Response
ML;	MIX LIMIT = 15
ML 4;	MIX LIMIT CHANGED FROM 15 TO 4

MM

The MM system command sets certain attributes of the MCP Memory Management System.

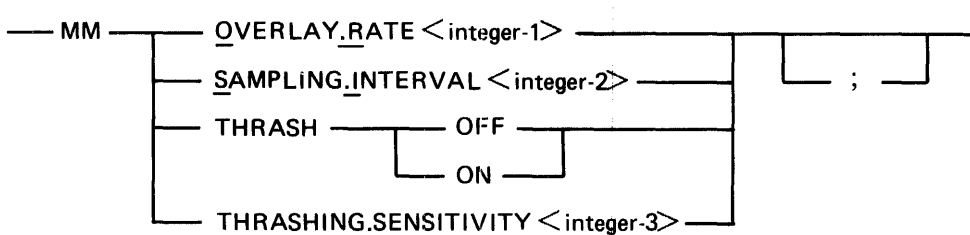
The MM system command is not allowed if the Second-Chance, Memory Management algorithm is used, that is, if the THR and MPRI MCP options are not set.

The values assigned to all the attributes, except for the SAMPLING.INTERVAL attribute, are retained by the MCP when a clear/start operation is performed, and need not be specified once set.

If <integer> is omitted for the OVERLAY.RATE, THRASHING.SENSITIVITY, or SAMPLING.INTERVAL attributes, or if the ON or OFF keywords are omitted for the THRASH attribute, the current value of the attribute is displayed.

Refer to appendix A, MCP Memory Management, for further operational details on Thrashing Detection and Priority Memory Management.

Remote ODT syntax:



Semantics:

OVERLAY.RATE

The OVERLAY.RATE keyword specifies the maximum overlay rate the system allows before reporting a thrashing condition. The default value is 10 overlays per second.

The coldstart programs (COLDSTART/DISK and COLDSTART/TAPE) set the OVERLAY.RATE to 10.

integer-1

This field, which must contain an integer in the range 1 to 20 inclusive, specifies the number of overlays per second allowed by the system before the SYSTEM IS THRASHING message is displayed. The default value is 10 and is set by the coldstart programs (COLDSTART/DISK and COLDSTART/TAPE).

SAMPLING.INTERVAL

The SAMPLING.INTERVAL keyword specifies the amount of time to wait before scanning memory. The default value is set by the MCP during the clear/start operation, and is dependent on the system memory size. This default value cannot be changed by the MM system command unless the DEBUG MCP option is set. Changes to the default value are not recommended.

MM

integer-2

This field must contain an integer in the range 1 to 50 inclusive. The value of <integer-2> is dependent on the system memory size and is set by the MCP during the clear/start operation. Changes to <integer-2> are not recommended.

THRASH

The THRASH keyword specifies the frequency that the MCP displays the SYSTEM IS THRASHING message when the thrashing condition has been detected by GISMO.

OFF

The OFF keyword is used with the THRASH keyword and causes the SYSTEM IS THRASHING message to be displayed as long as thrashing continues and only when a program enters or leaves the mix.

ON

The ON keyword is used with the THRASH keyword and causes the SYSTEM IS THRASHING message to be displayed at each N.SECOND interval (a variable period of time determined by the number of programs in the mix) as long as thrashing continues.

THRASHING.SENSITIVITY

The THRASHING.SENSITIVITY keyword specifies the maximum number of seconds that the overlay rate can be continuously exceeded before the MCP receives a thrashing interrupt.

integer-3

This field, which must contain an integer in the range 10 to 60 inclusive, specifies the maximum number of seconds that the overlay rate can be continuously exceeded before the SMCP receives a thrashing interrupt. The default is 20 and is set by the coldstart programs (COLDSTART/DISK and COLDSTART/TAPE).

Examples:

Command	Response
MM OVERLAY.RATE;	OVERLAY.RATE = 10
MM THRASH ON;	THRASHING MESSAGE ON
MM THRASHING.SENSITIVITY 15;	THRASHING.SENSITIVITY = 15

MP (Memory Priority)

The MP system command interrogates or changes the MEMORY.PRIORITY attribute of a program currently in the mix.

The MP system command is valid only when the MPRI MCP option is set.

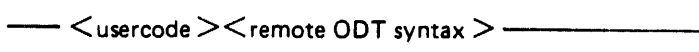
If <integer> is omitted, the MCP displays the current value of the MEMORY.PRIORITY attribute for the specified program.

Refer to the MEMORY.PRIORITY program control instruction in section 4 for further information on MEMORY.PRIORITY.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a valid mix number of a program currently in the mix, specifies the task in which to interrogate or change the MEMORY.PRIORITY attribute.

integer

This field, which must contain an integer in the range 0 to 15 inclusive, specifies the value of the MEMORY.PRIORITY attribute for the program specified by <mix-number>.

Examples:

Command	Response
1245MP =;	NDL/HANDLER = 1245 MP = 15
1269MP 7;	TEST/PROGRAM = 1269 MEMORY PRIORITY CHANGED TO 7

MR (Close Output File with Purge)

The MR system command causes the MCP to save the old file by purging the newly created file when a duplicate file situation occurs.

Remote ODT syntax:

— <mix-number>MR —┐
└───┬───┘
└───┬───┘
 ;

Workstation syntax:

— <usercode><remote ODT syntax> —┐
└───┬───┘

Semantics:

mix-number

This field must contain a mix number of a program waiting to close a file for which another file with the same name exists on disk.

Example:

1256MR;

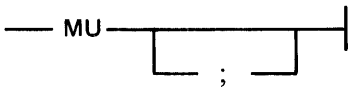
MU (List Multipack File Tables)

The MU system command interrogates the multipack file information table maintained by the MCP. This table contains information concerning all multipack files currently in use on the system, as well as all multipack files that were in use when the system was interrupted or halted and a clear/start operation was required.

The MCP removes the information in the multipack file information table when the user count for the file goes to zero, that is, all programs using the file have gone to end of job.

An entry is retained in the multipack file information table for any multipack file that is still in use when a clear/start operation occurs, until explicitly deleted from the table by the system operator. Refer to the RT system command for information about deleting entries from the multipack information table. Because such a file can be only partially created, yet entered into the disk directory on one or more user disks, the multipack file information table provides the system operator with the serial numbers of all user disks that can require manual action; for example, mounting the user disk and removing the partially-created file.

Remote ODT syntax:



Examples:

Command	Response
MU;	NO MULTI PACK FILES IN USE
MU;	"USERPACK/TEST/FILE" BASE = 123456 CONTINUATION PACK SERIAL NUMBER = 654321 OFF LINE. CONTINUATION PACK SERIAL NUMBER = 555555 ON LINE. CONTINUATION PACK SERIAL NUMBER = 121212 OFF LINE.
MU;	"PAYROLL/MASTER/" BASE = 111111 WAS IN USE LAST CLEAR/START CONTINUATION PACK SERIAL NUMBER = 222222 ON LINE. CONTINUATION PACK SERIAL NUMBER = 333333 ON LINE.

MX (Mix and Status Interrogation)

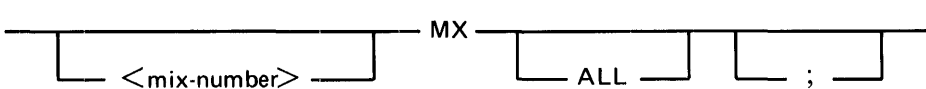
The MX system command displays the priority and current status of one or all programs in the mix. The MX system command functions identically to the WY system command.

If the MX system command is prefaced with a mix number, only information about that specific program is displayed. If the mix number is omitted, information about all visible tasks in the mix is displayed. Refer to the IV system command for information concerning task invisibility. If the optional ALL keyword is included, information about all tasks, including invisible tasks, is displayed.

If a program is awaiting operator action, the information displayed includes the action alternatives available to the operator.

If the MPRI MCP option is set, the processor and memory priorities are displayed instead of the priority number.

Syntax:



Semantics:

mix-number

This field, which must contain a valid mix number of a program currently in the mix, specifies the task in which to interrogate the priority and status.

ALL

The ALL keyword specifies that information about all tasks, including invisible tasks, is to be displayed.

The following examples assume that the MPRI MCP option is set. If the MPRI MCP option is not set, the priority number is displayed instead of the processor and memory priorities.

Examples:

Command	Response
MX;	SMCS = 308 PP=13, MP=13 "WAIT" STATUS CANDE = 321 PP=12, MP=12 "WAIT" STATUS RD = 310 PP=11, MP=11 "WAIT" STATUS SYCOM : PS =311 PP=9, MP=9 "WAIT" STATUS 6 PROGRAMS RUNNING...END MX/WY.
MX ALL;	SYSTEM/ODT = 304 PP=15, MP=15 EXECUTING GEM/NC__15FEB = 303 PP=15, MP=15 EXECUTING SMCS = 308 PP=13, MP=13 "WAIT" STATUS CANDE = 321 PP=12, MP=12 "WAIT" STATUS RD = 310 PP=11, MP=11 "WAIT" STATUS SYCOM : PS =311 PP=9, MP=9 "WAIT" STATUS 6 PROGRAMS RUNNING...END MX/WY.
321MX;	CANDE = 321 PP=12, MP=12 "WAIT" STATUS

NC (Network Controller)

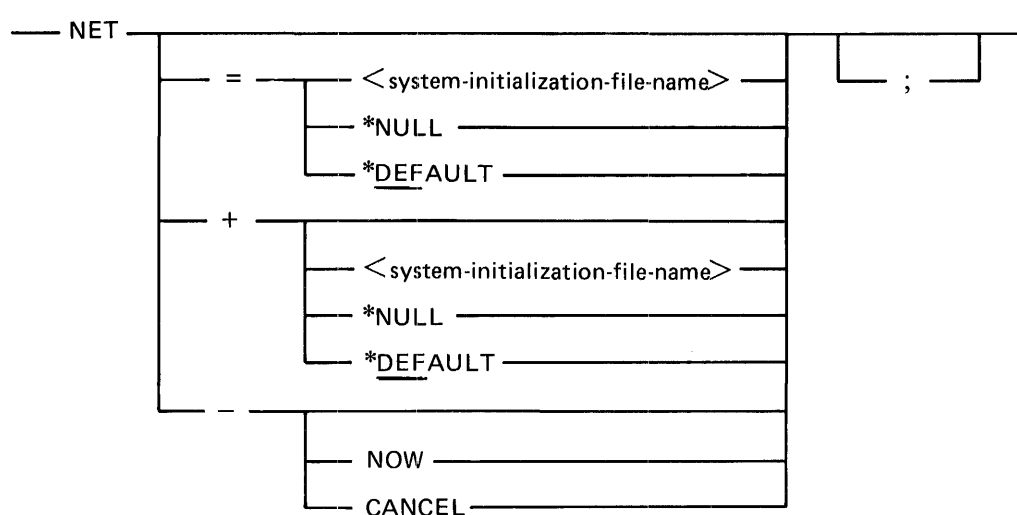
The NC system command communicates with the network controller. Refer to section 6 for the complete description of the NC system command.

NET (Network Mode Change or Inquiry)

The NET system command interrogates or changes the mode of the local B 1000 BNA system. Entering the NET system command without any additional options causes the current mode of the local B 1000 BNA system to be displayed.

Refer to the B 1000 Systems Burrough Network Architecture (BNA) Installation and Operation Manual, for a complete description of the B 1000 BNA system.

System ODT syntax:



Semantics:

system-initialization-file-name

This field, which must contain a valid B 1000 filename, specifies the initialization file to be read.

CANCEL

The CANCEL keyword causes the B 1000 BNA system to ignore the previous NET - system command and return to network mode.

***DEFAULT**

The *DEFAULT keyword changes the initialization file back to the default file name of BNA/NWINIT and uses that file name for initialization information.

NOW

The NOW keyword causes an immediate shutdown of the B 1000 BNA system. If the NOW keyword is not specified, the B 1000 BNA system waits for any open ports to close and prevents any further traffic.

***NULL**

The *NULL keyword inhibits the use of an initialization file. Some other agent must supply the required commands for the initialization of the node into network mode.

Examples:

NET + BNA/INIT.FILE;

NET - NOW;

NET;

NW (Network Message)

The NW system command causes a message to be routed to the Network Services Manager (BNA/NSM) program.

Remote ODT syntax:

— NW <message> ————
 └───┬───┘
 ; └───┘

Semantics:

message

This field must contain a valid BNA command and is routed to the BNA/NSM program. Refer to the B 1000 System Burroughs Network Architecture (BNA) Installation and Operation Manual for a complete description of the valid BNA commands.

Example:

NW ADD STATION NEWHIRE

OF (Optional File Response)

The OF system command informs the MCP that the specified file is optional and can be bypassed when a no-file condition occurs for a program with an optional file.

A optional file is an input file that has the OPTIONAL file attribute specified.

Remote ODT syntax:

— <mix-number> OF ————
 └───┬───┘
 └───┬───┘
 ; —┬—
 └──┘

Workstation syntax:

— <usercode> <remote ODT syntax> ————┘

Semantics:

mix-number

This field must contain a mix number of a program waiting on a no-file condition and the file has the OPTIONAL file attribute specified.

Example:

MCP Message	Command
INVESTMENT/FORCAST = 8874 FILE CARDS (LABELED "SUPPLEMENT") NOT PRESENT	8874OF;

OK (Continue Processing)

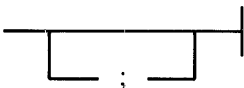
The OK system command causes the specified program to continue processing if the program was marked as WAITING.

The OK system command is used following operator correction of the following conditions.

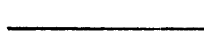
- DUPLICATE INPUT FILES
- DUPLICATE FILE ON DISK
- NO DISK
- DUPLICATE DATA DECKS
- FILE <file-identifier> NOT PRESENT
- STOPPED (ST) PROGRAMS

If the corrective action is not taken before the OK system command is entered, the original MCP message is repeated.

Remote ODT syntax:

— <mix-number> OK — 

Workstation syntax:

— <usercode><remote ODT syntax> — 

Semantics:

mix-number

This field must contain a mix number of a program that is currently marked as WAITING and for which action has been taken to correct the problem.

Examples:

MCP Message	Command
DMPALL =4573 DUPLICATE INPUT FILE "MASTER"	4573OK;
MTC SAVED	4573OK;

OL (Display Peripheral Status)

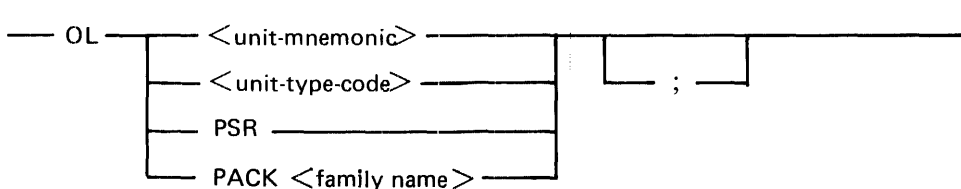
The OL system command interrogates the status of peripheral units.

If an invalid <unit-type-code> is specified, or if no units of the specified type exist on the system, the MCP displays the following message.

```
NULL <unit-type-code> TABLE
```

For disk packs and cartridges, if a program is readying the pack, the mix number and the name of that program is reported.

Syntax:



Semantics:

unit-mnemonic

This field must contain a valid unit mnemonic and causes the status of the specified unit to be displayed. For disk devices, the number of active users for the unit is also displayed.

unit-type-code

This field must contain the initial characters of a unit mnemonic and causes the status of all the peripherals of the same type to be displayed.

The following are the valid <unit-type-codes>.

- CD - Data Recorders
- CP - Card Punch only
- CR - Card Reader only
- CS - Cassette
- DC - Disk Cartridge
- DK - Head-per-track Disk (5N)
- DP - Disk Pack
- FD - Floppy Disk
- LP - Line Printer
- MLC - Multiline Control
- MT - Magnetic Tape
- PP - Paper Tape Punch
- PR - Paper Tape Reader
- SLC - Single-Line Control
- SR - Reader Sorter

PSR

The PSR keysymbol causes the current status of the pseudo readers to be displayed.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

Examples:

Command	Response
OL CDA;	CDA NOT READY
OL MTC;	MTC UNLABELED
OL DPA;	DPA LABELED "USER" (U) #123456
OL DPB	DPB LABELED "USERPACK" (U) #123456 BEING READYIED BY SYSTEM/PANDA (123)
OL MTA	MTA LABELED "MASTER" [123456]
OL LP	LPA AVAILABLE FOR OUTPUT. LPB NOT READY

OU (Specify Output Device)

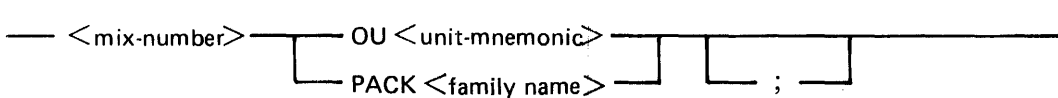
The OU system command directs an output file to the specified output device.

The OU system command is used in response to the PUNCH REQUIRED ... or PRINTER REQUIRED... message to direct the file to backup.

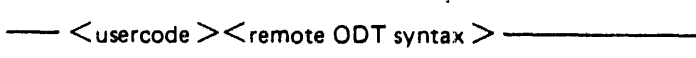
The OU system command can also be used in a special-forms-required condition, and functions exactly the same as the FM system command. In this case, the OU system command overrides a saved condition on the specified device. This allows the system operator to mount the special form before it is actually required and save the device with the SV system command to prevent its use by other programs in the mix.

When a file with the SPECIAL FORMS file attribute set is closed, the device to which it is assigned is saved automatically by the MCP, thus preventing other programs from accidentally using the special form before it can be removed by the operator.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field must contain a mix number of a program that has an output file that is opened and that requires an output device.

unit-mnemonic

This field, which must contain a valid output device unit mnemonic, specifies the unit to use for the output file that requires an output device.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of a unit mnemonic.

Examples:

1219OU LPA;

2317OU MTC;

PASSWORD

PASSWORD

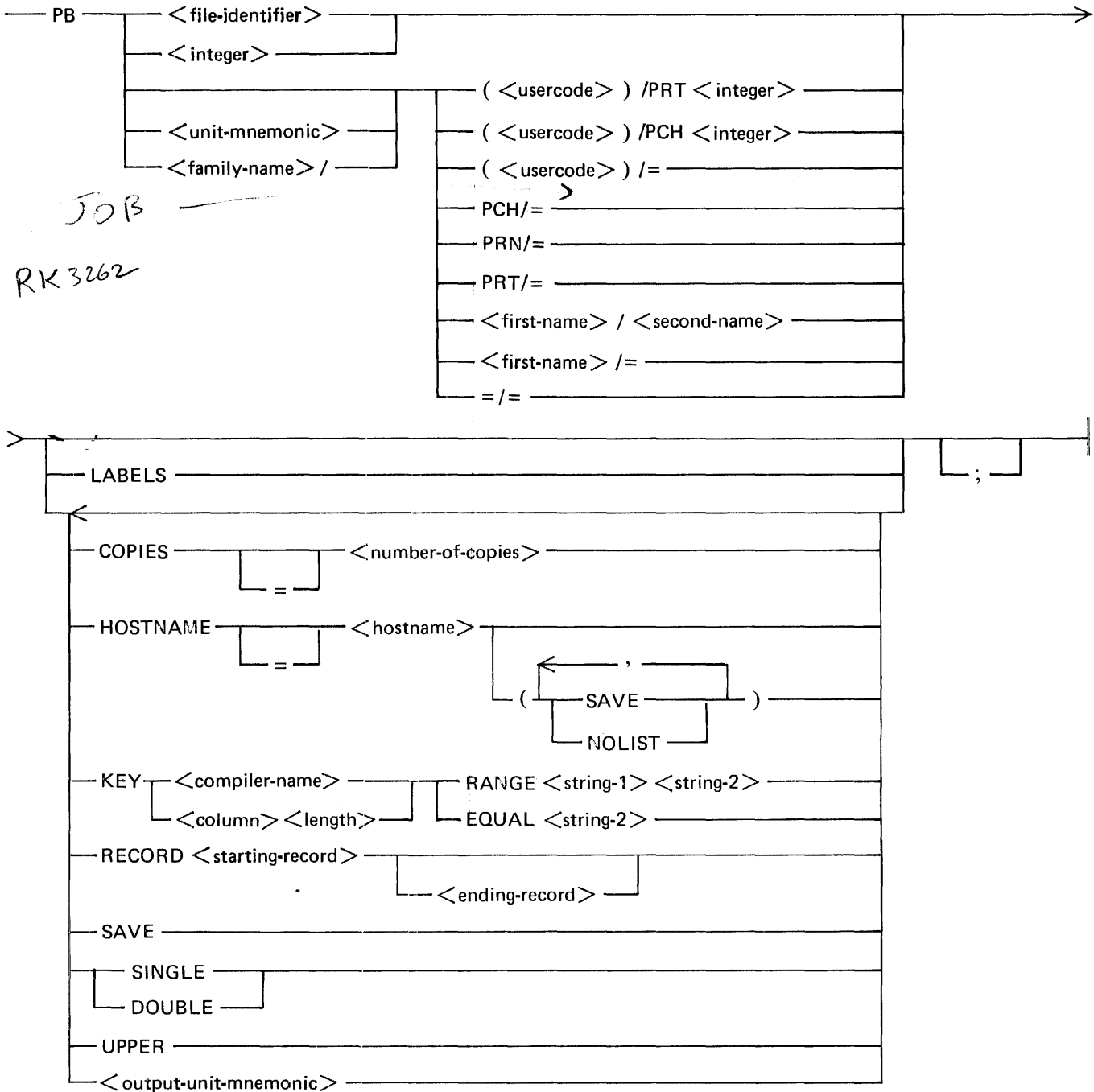
The **PASSWORD** command permanently changes the password associated with a usercode. The **PASS-WORD** command requires that the command end with the end of text, that is, another command cannot follow the **PASSWORD** command.

Refer to the B 1000 Systems Work Flow Language (WFL) Language Manual for more information.

PB (Print/Punch Backup)

The PB system command initiates the SYSTEM/BACKUP program to print or punch a disk or tape backup file.

Remote ODT syntax:



Semantics:

compiler-name

This field must contain a valid compiler name and causes the automatic generation of the proper column number and length pair that corresponds to the sequence number field of the output listing produced by the compiler specified.

column

This field, which must contain an integer in the range 1 to 132 inclusive, specifies the beginning column number of the subfield to be used for the compare argument when the KEY keyword is specified.

family-name

This field must contain a valid disk identifier of a user disk that contains the backup file to be printed or punched.

ending-record

This field must contain a relative record number of a record in the backup file and specifies the last record to be printed. Printing begins with the relative record number specified by <starting-record>. This field is used in conjunction with the RECORD keyword.

file identifier

Refer to file identifier in section 1 for a complete description of file identifier.

first-name

This field, which must contain a valid first name, specifies the first part of the file name of the backup file to be printed or punched.

second-name

This field, which must contain a valid file identifier, specifies the subdirectory name of the backup file to be printed or punched.

hostname

This field, which must contain a valid BNA host name, specifies the host system in which the backup file is to be printed. This field is used in conjunction with the HOSTNAME keyword.

length

This field, which must contain an integer in the range 0 to 10 inclusive, specifies the length of the subfield to be used for the compare argument when the KEY keyword is specified.

number-of-copies

This field, which must contain an integer, specifies the number of copies of the backup file is to be printed or punched.

output-unit-mnemonic

This field, which must contain a valid line printer or card punch unit mnemonic, specifies the unit to which the output is to be directed.

starting-record

This field, which must contain a valid relative record number within the backup file, specifies the record in which to begin printing or punching the file. This field is used in conjunction with the RECORD keyword.

string-1

This field, which must contain a string of characters having a maximum length of `<length>`, is specified with the RANGE keyword. Printing or punching begins when an exact comparison is made between the subfield and `<string-1>`. Printing or punching continues until an exact comparison is made between the subfield and `<string-2>`, or until the end of file is reached.

string-2

This field, which must contain a string of characters having a maximum length of `<length>`, is specified with the RANGE keyword. Printing or punching begins when an exact comparison is made between the subfield and `<string-1>`. Printing or punching continues until an exact comparison is made between the subfield and `<string-2>`, or until the end-of-file record is read.

string-3

This field, which must contain a string of characters having a maximum length of `<length>`, is specified with the EQUAL keyword. Printing or punching begins when an exact comparison is made between the subfield and `<string-3>` and continues until the end-of-file record is read.

unit-mnemonic

This field, which must contain a valid magnetic tape or user disk unit mnemonic, specifies the output device to which the backup print or punch file is to be directed.

usercode

This field, which must contain a valid usercode, specifies the first name portion of the backup file name in which to print or punch the file.

COPIES

The COPIES keyword causes the SYSTEM/BACKUP program to print or punch the number of copies of the backup file specified by `<number-of-copies>`.

DOUBLE

The DOUBLE keyword causes the SYSTEM/BACKUP program to double space the entire printer listing, overriding any carriage control specified in the backup printer file.

EQUAL

The EQUAL keyword causes the SYSTEM/BACKUP program to begin printing or punching the backup file when an exact comparison is made between the subfield and `<string-3>`, and continues printing or punching the backup file until the end-of-file record is read.

HOSTNAME

The HOSTNAME keyword causes the SYSTEM/BACKUP program to transfer the backup file to the BNA host system with `<hostname>` as the host name.

KEY

The KEY keyword causes the SYSTEM/BACKUP program to print or punch a range of records according to information within the backup-file records; for example, sequence number. The portion of each record to be compared can be specified, as well as the information that starts and stops the output.

LABELS

The LABELS keyword causes the SYSTEM/BACKUP program to print the label record of the backup file.

NOLIST

The NOLIST keyword causes the SYSTEM/BACKUP program to notify the receiving BNA host system that the backup file is not to be printed or punched.

PCH/=

The PCH/= key symbol causes the SYSTEM/BACKUP program to punch all the backup punch files with the default punch backup file name of BACKUP/PCH<integer>.

PRN/=

The PRN/= key symbol causes the SYSTEM/BACKUP program to print all the backup print files with the default printer backup file name of BACKUP/PRT<integer>.

PRT/=

The PRT/= key symbol causes the SYSTEM/BACKUP program to print all the backup print files with the default printer backup file name of BACKUP/PRT<integer>.

RANGE

The RANGE keyword causes the SYSTEM/BACKUP program to begin printing or punching when an exact comparison is made between the subfield and <string-1>. The printing and punching continues until an exact comparison is made between the subfield and <string-2>, or until the end-of-file record is read.

RECORD

The RECORD keyword causes the SYSTEM/BACKUP program to print a range of records beginning with <starting-record> and ending with <ending-record>. If <ending-record> is not specified, all records from <starting-record> to the end of the file are printed. If the RECORD keyword is not specified, all records in the file are printed. The RECORD keyword is not valid when an equal sign (=) character is used as part of a file specifier.

SAVE

The SAVE keyword is used in two places in the PB system command syntax.

If the SAVE keyword is specified in the HOSTNAME portion of the PB system command syntax, the SYSTEM/BACKUP program notifies the receiving BNA host system to save the backup file. The receiving BNA host system makes the final determination of whether or not to save the backup file.

If the SAVE keyword is specified after the backup file name, the SYSTEM/BACKUP program does not remove the file after the backup file is printed or punched.

SINGLE

The SINGLE keyword causes the SYSTEM/BACKUP program to single space the entire backup printer file, overriding any carriage control instruction in the backup printer file.

UPPER

The UPPER keyword causes the SYSTEM/BACKUP program to translate all lower-case characters to upper-case characters before printing.

=/=

The =/= key symbol causes all the backup files with the default printer backup name of BACKUP/PRT<integer> to be printed and all backup files with the default backup name of BACKUP/PCH<integer> to be printed.

=

The = key symbol is optionally used with the COPIES and HOSTNAME keywords.

NOTE

If both the RECORD and KEY keywords are specified, the comparisons specified by the KEY keyword are made only in the range of records specified by the RECORD keyword.

Examples:

PB 125;

PB 17 LPA SAVE;

PB DCC 4 RECORD 5;

PB MTA =/=;

PB 3 KEY COBOL RANGE 123 567;

PB 2 KEY 7 6 EQUAL "ABC";

PB 53 RECORD 1 100 DOUBLE SAVE;

PB 24 SAVE HOSTNAME THERE (SAVE);

PB PRT/= COPIES=2 LPB;

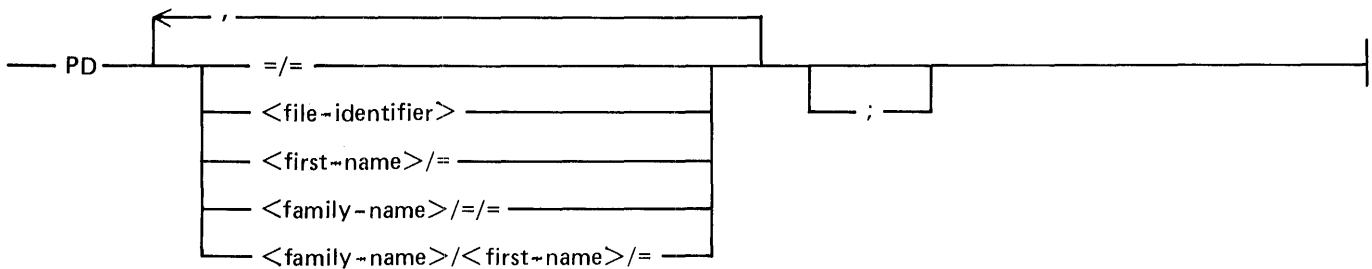
PD (Display Directory)

The PD system command causes the specified files to be displayed or interrogates a disk directory for a specific file(s).

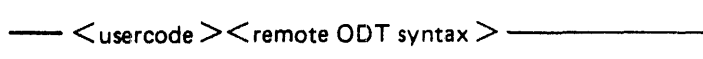
If the file name is not present in the disk directory, the MCP responds with the message: <file-identifier> NOT IN DIRECTORY.

The specific commands PD =/= and PD <family-name>/=/= cause the SYSTEM/PANDA program to execute and print an alphabetized printer listing of the files. These two commands are not allowed if the SYSTEM/PANDA program is not present.

Remote ODT syntax:



Workstation syntax:



Semantics:

family-name

This field, which must contain a disk identifier, specifies the user disk on which to interrogate the disk directory.

first-name

This field, which must contain a first name, specifies the first part of a file name.

=/=

The =/= keysymbol causes all the files on the user disk or SYSTEM disk to be displayed if this command is issued under a usercode. Otherwise, the SYSTEM/PANDA program is called to do the PD. In this case, if the SYSTEM/PANDA program is not present, then an error message is displayed.

Examples:

Command	Response
PD COBOLZ;	COBOLZ NOT IN DISK DIRECTORY END PD.
PD FORTRAN77;	PD= FORTRAN77 END PD.
PD USER/MASTER/=;	PD= USER/MASTER/FILE PD= USER/MASTER/TEST PD= USER/MASTER/PROG END PD.
PD DMPALL, COBOL74, RPG;	PD= DMPALL PD= COBOL74 PD= RPG END PD.

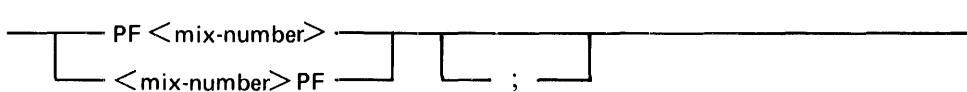
PF (Print Fetch)

The PF system command displays messages from a task that contains a Work Flow Language (WFL) FETCH specification and is awaiting a fetch action. Such tasks are entered in the waiting schedule awaiting an OK system command and the following message is displayed.

JOB <mix-number> CONTAINS FETCH MESSAGE; "PF" REQUESTED

The PF system command can then be entered or can be bypassed by entering the OK system command. The OK system command makes the task eligible to be selected for execution.

Syntax:



Semantics:

mix-number

This field must contain a mix number of a task in the waiting schedule that requires a fetch action.

Example:

Command	Response
2391 PF	2391 FETCH: SAMPLE FETCH MESSAGE.

PG (Purge)

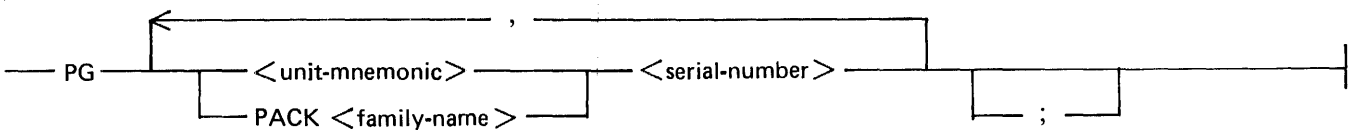
The PG system command purges a user disk or magnetic tape.

A purged user disk is marked as UNRESTRICTED and its disk identifier remains unchanged.

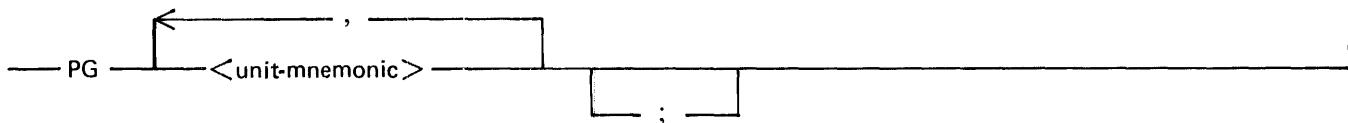
A magnetic tape must have a write ring in place in order to be purged.

The serial number of a magnetic tape is not changed when the magnetic tape is purged. To assign or change the serial number of magnetic tape, use the SN system command.

Remote ODT disk syntax:



Remote ODT tape syntax:



Semantics:

unit-mnemonic

This field must contain a user disk or magnetic tape unit mnemonic. This mnemonic specifies the unit to be purged.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of a unit mnemonic.

serial-number

This field must contain the six-digit serial number of the user disk to be purged. This field is not used for magnetic tape.

Examples:

PG DPA 000456;

PG MTC, MTD;

NOTE

Do not enter the PG system command for a pack that is in use. Check the user count of the pack with the OL system command, and consider the programs running in the mix. They must not be using the pack in any way.

PM (Print Memory Dump)

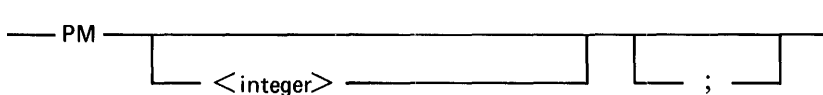
The PM system command allows a system operator to analyze and print a program dump file or package a system dump file.

A program dump has the name DUMPFIL/ <integer> or (<usercode>)/DMP <integer>. When the PM system command is entered with an <integer> as a parameter, the SYSTEM/IDA program is initiated. The SYSTEM/IDA program analyzes and prints the contents of the dump file.

A system dump file has the name SYSTEM/DUMPFIL. If the PM system command does not include an <integer>, the SYSTEM/IDA program is initiated. The SYSTEM/IDA program packages the contents of SYSTEM/DUMPFIL. The packaged dump file has the name DUMPFIL/PM <nnn>; however, the operator can file-equate the internal file PM to a different name when the PM system command is entered.

The SYSTEM/IDA programs must be located on the SYSTEM disk to use the PM system command.

Remote ODT syntax:



Workstation syntax:



Semantics:

integer

This field, which must contain an integer, specifies the number of the program dump file to be analyzed. The dump file must have the name DUMPFIL/<integer>.

SYSTEM/IDA

Switch	Value	Result
0	0	Analyze only the first 64 elements of each array.
	1	Analyze all elements of each array.
	2	Do not analyze array elements.
1	0	Allow comparison of resident code, interpreter, and microcode segments to their copies on disk.
	1	Suppress code segment comparison.
2	0	Display DMS and ISAM buffers.
	1	Suppress DMS and ISAM buffer data only.
	2	Suppress DMS and ISAM buffer descriptors and data.
3	0	Suppress printing of certain (already analyzed) memory areas during hexadecimal dump.
	1	Print all areas of memory during hexadecimal dump.
4	0	Remain in loop until exit conditions are satisfied.
	1	Decrement switch 4 and exit loop unconditionally.

Switch	Value	Result
5	0	Analyze all ODT queue entries.
	1	Analyze only the last 25 percent of the ODT queue.
6	0	Print available memory areas during hexadecimal dump.
	1	Suppress printing of available memory areas.
7		Reserved for development use.
8	0	Abort analysis if the MCP level is incorrect.
	1	Attempt system dump analysis regardless of MCP level
9	0	Default option UPPERCASE = ON.
	1	Default options UPPERCASE and DOUBLESPEACE = OFF
	2	Default options UPPERCASE and DOUBLESPEACE = ON
	3	Default option DOUBLESPEACE = ON

Examples:

PM

PM 3459

PO

PO (Power Off)

The PO system command informs the MCP that a user disk or Phase-Encoded (PE) tape is to be removed from the system.

The PO system command is invalid for a SYSTEM disk.

When B 9484 or B 9486 disk drives are in use, the MCP attempts to physically power down the unit requested. When B 9495 or B 9496 tape units are attached to a B 1394 or B 1495 tape control, the tape is unloaded if it is not in use. The PO system command is ignored for other types of units.

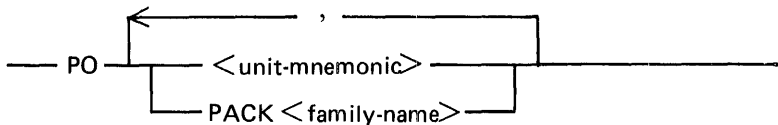
If a PO system command is entered for a user disk currently in use, the following message is displayed:

```
<unit-mnemonic> HAS <integer> USERS - CANNOT "PO"
```

This means that a request to power off a pack with a user count greater than 0 is rejected unless the NO.USERS option is specified. When the NO.USERS option is specified, the the system accepts the PO command for packs with a user count greater than or equal to 0. The pack is powered off when the user count becomes 0.

The PO system command can be used on a multipack file base pack if there are no single-pack files in use at the time of the request.

Remote ODT syntax:



Semantics:

unit-mnemonic

This field, which must contain a valid tape or user disk unit mnemonic, specifies the unit to power off.

PACK

The PACK keyword specifies that a disk identifier follows instead of a unit mnemonic.

family-name

This field, which must contain a valid name of a user disk, specifies the disk unit to power off.

Examples:

Command	Response
PO DPB;	DPB IS NOW OFFLINE
PO MTC;	(No response is generated for magnetic tape)
PO PACK PAYROLL	DPC IS NOW OFFLINE

PP (Processor Priority)

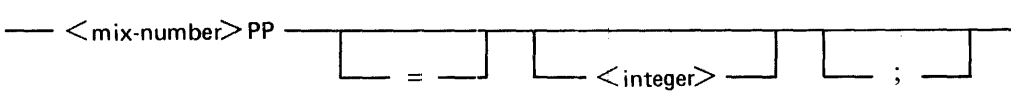
The PP system command interrogates or changes the PROCESSOR.PRIORITY of a program currently in the mix.

The PP system command is valid only when the MPRI MCP option is set.

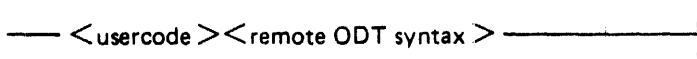
If <integer> is omitted, the MCP displays the current value of the PROCESSOR.PRIORITY for the specified program.

Refer to the PROCESSOR.PRIORITY program control instruction in section 4 for further information on PROCESSOR.PRIORITY.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of a program currently in the mix, specifies the program to interrogate or that the PROCESSOR.PRIORITY attribute is to be changed.

integer

This field, which must contain an integer in the range 0 to 15 inclusive, specifies the value of the PROCESSOR.PRIORITY attribute. The default value is 4.

Examples:

Command	Response
9203PP;	DMPALL =9203 PP=7
1276PP 12;	NDL/HANDLER =1276 PROCESSOR PRIORITY CHANGED TO 12

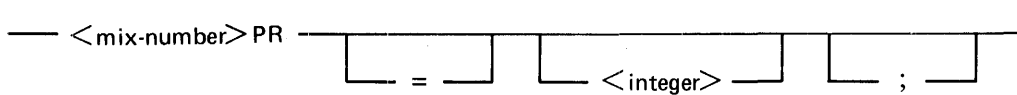
A handwritten signature or initials, possibly 'RW', located in the lower right quadrant of the page.

PR (Change Priority)

The PR system command changes the PROCESSOR.PRIORITY attribute of a program currently in the mix. This command can also be used to interrogate the Processor and Memory priorities of a program.

If <integer> is omitted, the MCP displays the current value of the MEMORY.PRIORITY and PROCESSOR.PRIORITY attributes for the specified program. If the MPRI MCP option is reset, these two priority values are considered to be the same and are displayed as one value labeled PR.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of a program currently in mix, specifies the program for which the PR system command is to be applied to.

integer

This field, which must contain an integer in the range 0 to 15 inclusive, specifies the new value of the PROCESSOR.PRIORITY attribute for the program.

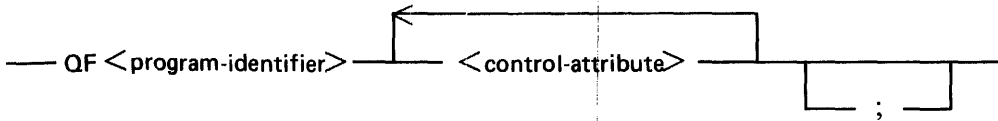
Examples:

Command	Response
1987PR;	DMPALL = 1987 PP = 7, MP = 7
2987PR 13;	NDL/HANDLER = 2987 PRIORITY CHANGED TO 13

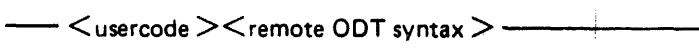
QF (Query File)

The QF system command is used to interrogate a program on disk concerning the status of its control attributes.

Remote ODT syntax:



Workstation syntax:



Semantics:

program-identifier

This field, which must contain a program name of a program that currently exists in the disk directory, specifies the name of the program to interrogate.

control-attribute

This field, which must contain a valid program control instruction, specifies the control attribute to interrogate. Refer to section 4 for a description of program control instructions.

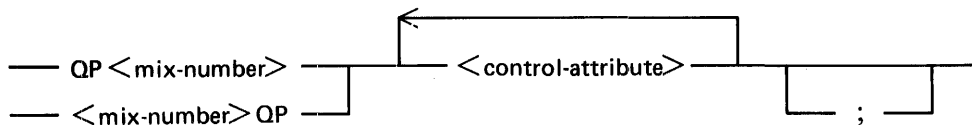
Examples:

Command	Response
QF DMPALL CG;	QF DMPALL : CHARGE NUMBER = 999999
QF RPG FILE LINE BACKUP;	QF RPG : IN "LINE" BACKUP.STATUS = HARDWARE OR BACKUP.DISK OR BACKUP.TAPE

QP (Query Program)

The QP system command interrogates an executing or scheduled program for the status of its control attributes.

Syntax:



Semantics:

mix-number

This field, which must contain a mix number of an executing or scheduled program, specifies the program to interrogate the control attributes.

control-attribute

This field, which must contain a valid program control instruction, specifies the control attribute to interrogate. Refer to section 4 for a description of program control instructions.

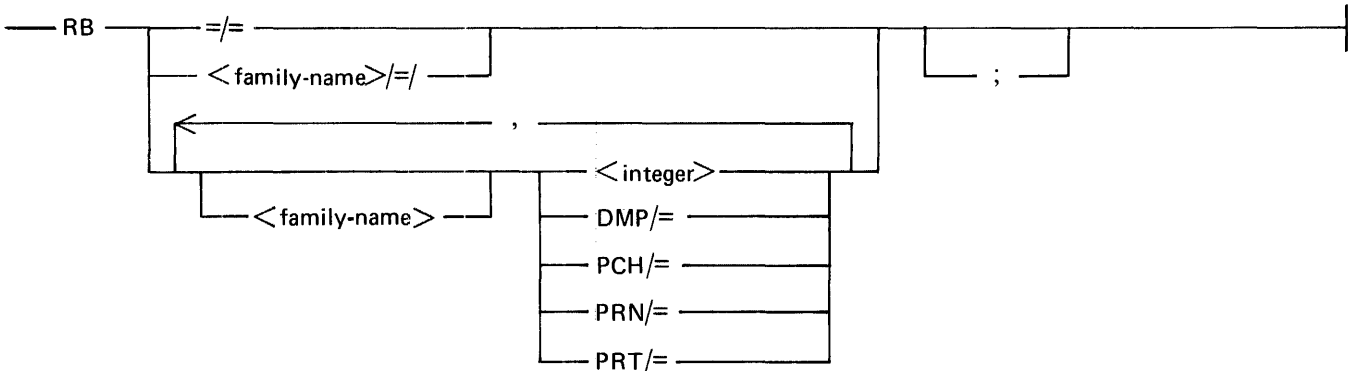
Examples:

Command	Response
QP 14 CHARGE FREEZE	QP DMPALL : CHARGE = 31404 QP DMPALL : FREEZE FLAG = 0
QP 15 PRIORITY;FILE LINE LAB;	QP SYSTEM/COMPARE : PROCESSOR.PRIORITY = 5 QP SYSTEM/COMPARE : MEMORY.PRIORITY = 4 QP SYSTEM/COMPARE : IN "LINE" LABEL.TYPE = UNLABELED
16 QP MEMORY;	QP CANDE : DYNAMIC MEMORY SIZE = 5760

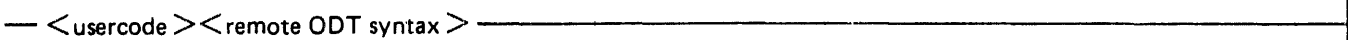
RB (Remove Backup Files)

The RB system command removes backup and memory dump files from disk. The RB system command functions identically to the RF system command.

Remote ODT syntax:



Workstation syntax:



Semantics:

family-name

This field, which must contain a valid disk identifier, specifies the user disk which contains the backup files to be removed.

integer

This field, which must contain an integer, specifies the backup file number to be removed. The backup file must have the default name of `BACKUP/PRT<integer>`, `BACKUP/PCH<integer>`, or `DUMPFIL/ <integer>` in order to be removed.

DMP/=

The `DMP/=` key symbol causes all the dump files with the default name of `DUMPFIL/ <integer>` to be removed.

PCH/=

The `PCH/=` key symbol causes all the punch files with the default name of `BACKUP/PCH<integer>` to be removed.

PRN/=

The `PRN/=` key symbol causes all the printer files with the default name of `BACKUP/PRT<integer>` to be removed.

PRT/=

The `PRT/=` key symbol causes all the printer files with the default name of `BACKUP/PRT<integer>` to be removed.

=/=

The `=/=` key symbol causes all the dump, printer, and punch backup files to be removed. The backup files must have the default name of `DUMPFIL/ <integer>`, `BACKUP/PCH<integer>`, and `BACKUP/PRT<integer>` in order to be removed.

Examples:

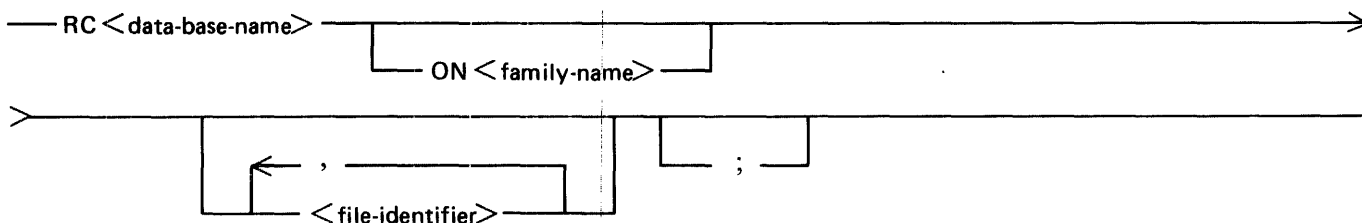
Command	Response
RB 37;	BACKUP/PRT37 REMOVED
RB =/=;	BACKUP/PRT12 REMOVED DUMPFIL/32 REMOVED BACKUP/PCH10 REMOVED
RB PCH/=;	BACKUP/PCH23 REMOVED BACKUP/PCH24 REMOVED
RB USER/=/=;	USER/BACKUP/PRT12 REMOVED
RB USER/= /215;	USER/DUMPFIL/215 REMOVED
USER SITE/A RB PRT/=;	(SITE)/PRT12 REMOVED (SITE)/PRT13 REMOVED

RC (Recover Database)

The RC system command is used in conjunction with DMSII audit and recovery to initiate recovery on a data base. The DMS/RECOVERDB program is initiated to recover the data base and must be present on the SYSTEM disk.

The list of <file-identifiers> allows a partial dump recovery to be performed on the specified DMSII files. This can be done if a subset of the data base, excluding the data base dictionary, has been lost or corrupted. Partial dump recovery requires that both a current copy of the dictionary and a backup copy corresponding to the files to be recovered be present on disk. The old dictionary must be labeled <data-base-name>/OLD.DICT.

Remote ODT syntax:



Semantics:

data-base-name

This field, which must contain a data base name, specifies the name of the DMSII data base to recover.

family-name

This field, which must contain a disk identifier, specifies the user disk on which the data base dictionary is located.

file-identifier

This field, which must contain a valid file identifier, specifies the name of the DMSII file on which to perform a partial dump recovery. If the file is located on the SYSTEM disk, the <file-identifier> must be in the form of <data-base-name>/<file-name>. If the file is located on a user disk, the <file-identifier> must be in the form of <family-name>/<data-base-name>/<file-name>.

Examples:

```
RC STUDENTDB;
```

```
RC PAYROLL ON USER1;
```

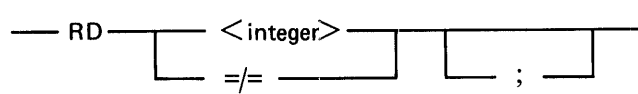
```
RC UNIV UNIV/FILE1, UNIV/FILE2, UNIV/FILE3;
```

```
RC UNIV ON USER USER/UNIV/FILE1, USER/UNIV/FILE2, USER/UNIV/FILE3;
```

RD (Remove Pseudo Card Files)

The RD system command removes pseudo-reader files from disk. If a pseudo-reader file is currently in use by a pseudo reader, the file is not removed.

Remote ODT syntax:



Semantics:

integer

This field, which must contain an integer, specifies the pseudo-reader file number to remove.

=/=

The =/= keysymbol causes all pseudo-reader files to be removed.

Examples:

Command	Response
RD 1;	DECK/1 REMOVED
RD =/=;	DECK/1 REMOVED DECK/2 REMOVED DECK/3 REMOVED

REMOVE

The REMOVE system command deletes specified files from the disk directory and makes the file space available to the MCP. The /= form deletes the main directory entry and all the files in its subdirectory.

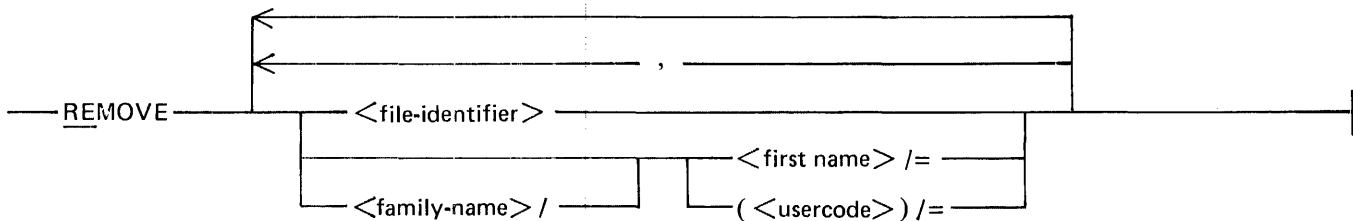
The following syntax diagram describes the Control Card syntax. When the WFL option is set, the WFL syntax is used. For the WFL syntax of the REMOVE ODT-command, refer to the B 1000 Systems WFL Reference Manual. For more information about the WFL option please refer to the WFL System Command or the WFL System Option in this document.

If <file-identifier> resides on a user disk, the disk identifier must precede the file name in order for the MCP to locate the correct file. When the disk identifier is not included, the MCP assumes that the file resides on the SYSTEM disk. One exception to this assumption occurs when the REMOVE system command is prefaced by the USER system command. In this case, the disk identifier for the REMOVE system command is taken from the (SYSTEM)/USERCODE file.

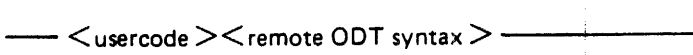
The REMOVE system command must be preceded by the USER system command with the required usercode/password combination when it is desired to remove PRIVATE files from disk.

If the REMOVE system command is entered by way of a card deck, additional cards can be included providing the last file to be removed has the semicolon (;) character following the REMOVE <file-identifier> statement.

Remote ODT syntax:



Workstation syntax:



Semantics:

family-name

This field, which must contain a valid B 1000 disk identifier, specifies the name of the user disk on which the file resides.

first-name

This field, which must contain a valid B 1000 first name, specifies the main directory name for the file.

REMOVE

Examples:

REMOVE A/B;

CC REMOVE USER/MASTER/FILE;

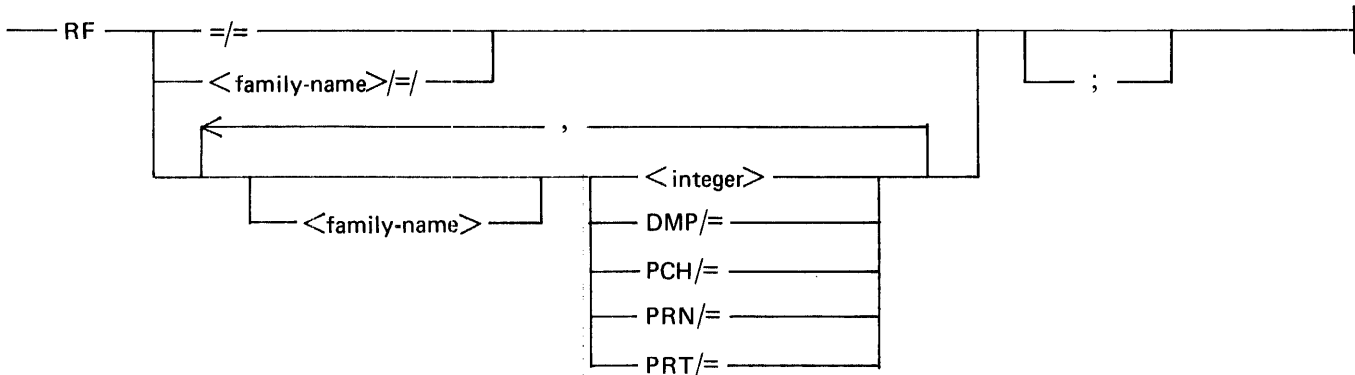
RE A, B, C, D;

USER FINANCE/ACCT RE FILE1, FILE2;

RF (Remove Backup Files)

The RF system command removes backup and memory dump files from disk. The RF system command functions identically to the RB system command.

Remote ODT syntax:



Workstation syntax:



Semantics:

family-name

This field, which must contain a valid disk identifier, specifies the user disk that contains the backup files to be removed.

integer

This field, which must contain an integer, specifies the backup file number to be removed. The backup file must have the default name of BACKUP/PRT<integer>, BACKUP/PCH<integer>, or DUMPFIL/<integer> in order to be removed.

DMP/=

The DMP/= key symbol causes all the dump files with the default name of DUMPFIL/<integer> to be removed.

PCH/=

The PCH/= key symbol causes all the punch files with the default name of BACKUP/PCH<integer> to be removed.

PRN/=

The PRN/= key symbol causes all the printer files with the default name of BACKUP/PRT<integer> to be removed.

PRT/=

The PRT/= key symbol causes all the printer files with the default name of BACKUP/PRT<integer> to be removed.

RF

=/=

The =/= keysymbol causes all the dump, printer, and punch backup files to be removed. The backup files must have the default name of DUMPFIL/ <integer>, BACKUP/PCH <integer>, and BACKUP/PRT <integer> in order to be removed.

Examples:

RF 37;

RF =/=;

RF USER/=/ =;

RF USER/=/215;

USER SITE/A RF PRT/=;

RL (Relabel User Disk)

The RL system command changes the disk identifier for a user disk.

The RL system command is not allowed on a user disk when it is assigned as a backup-designated user disk, when <family-name> is the same as a user disk currently on line on the system, or when the user count is greater than 0.

Remote ODT syntax:

```
RL <unit-mnemonic> <family-name>
   PACK <family-name> ;
```

Semantics:

unit-mnemonic

This field must contain a valid user disk unit mnemonic. The allowable unit mnemonics are DPx and DCx, where x can be any upper-case alpha character of the user disk to be changed.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of unit mnemonic.

family-name

This field, which must contain a valid disk identifier, specifies the new name for the user disk. This new name cannot be the same as a user disk currently on line or assigned the name DISK.

Examples:

```
RL DPB NEWUSER;
```

```
RL DCE NEWCART;
```

RM (Remove Duplicate Disk File)

The RM system command removes a disk file from the disk directory in response to a DUPLICATE FILE ON DISK message.

The DUPLICATE FILE ON DISK message is a result of a program attempting to close a disk output file with the same name as a file already in the disk directory. This causes the program to go into a waiting state. The RM system command causes the MCP to remove the old file, close the new file, enter the new file in the disk directory, and allow the program to continue processing.

Remote ODT syntax:

— <mix-number> RM ————
 └──┬──┘
 └──┬──┘
 ; —┘

Workstation syntax:

— <usercode> <remote ODT syntax > —————

Semantics:

mix-number

This field must contain a mix number of a program waiting resolution of a duplicate file situation.

Example:

421 RM;

RN (Assign Pseudo Readers)

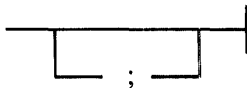
The RN system command assigns a specific number of pseudo readers and can be entered either before or after the creation of pseudo-reader files.

The operator must determine the optimum number of pseudo readers in relation to the number of pseudo-reader files to be processed.

By entering RN 0, all pseudo readers are closed as soon as they are finished processing the pseudo-reader files they are presently reading.

The pseudo-reader files are closed when a clear/start operation is performed.

Remote ODT syntax:

— RN <integer> — 

Semantics:

integer

This field, which must contain an integer, specifies the number of active pseudo readers to be used.

Example:

Command	Response
RN 1;	NUMBER OF PSEUDO READERS CHANGED FROM 0 TO 1.

RO (Reset Option)

The RO system command is used to reset MCP options.

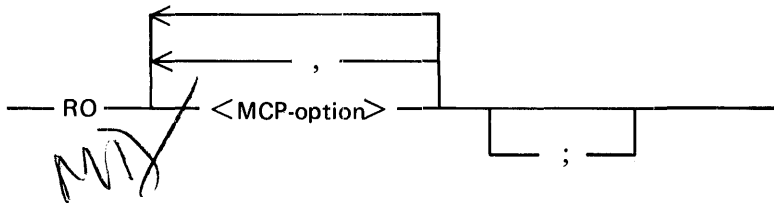
Certain MCP options cannot be reset using the RO system command. If an attempt is made to reset any of these options, the MCP displays the following message.

<MCP-option-name> LOCKED

The TO system command can be used to determine which options are currently set. The option indicator equals 1 when set and 0 when reset.

Refer to section 3 for a complete description of the MCP options.

Remote ODT syntax:



Semantics:

MCP-option

This field, which must contain a valid MCP option name, specifies the option to be reset. Refer to section 3 for a complete description of the valid MCP options.

Examples:

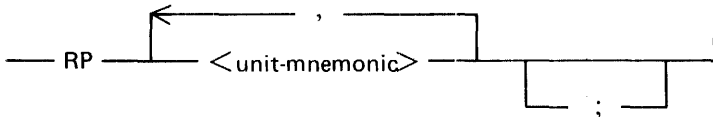
Command	Response
RO LIB;	LIB = 0
RO DATE, TIME;	DATE = 0 TIME = 0

RP (Ready and Purge)

The RP system command sets a tape unit in READY status and purges the tape unit.

The RP system command is only valid for magnetic tape.

Remote ODT syntax:



Semantics:

unit-mnemonic

This field, which must contain a valid magnetic tape unit mnemonic, specifies the tape drive to make ready and the tape to purge.

Examples:

Command	Response
RP MTA;	MTA SCRATCH [022381]
RP MTB, MTC;	MTB SCRATCH [013081]
	MTC SCRATCH [092881]

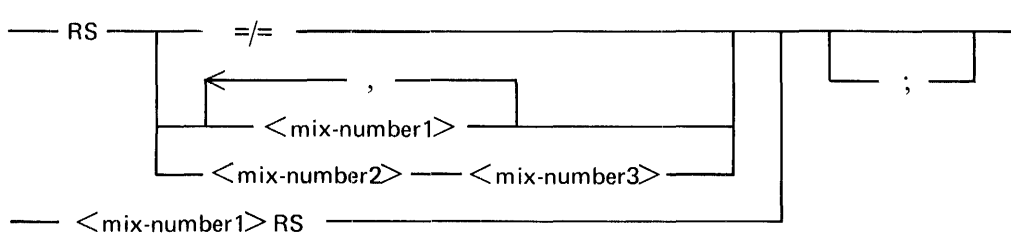
RS (Remove Jobs from Schedule)

The RS system command removes one or more tasks from the WAITING or ACTIVE SCHEDULE before it is entered in the mix for execution.

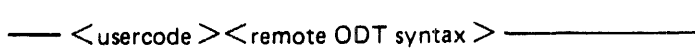
The RS system command can be used to remove a single task, a list of tasks, or a range of tasks from the WAITING or ACTIVE SCHEDULE.

If the requested task(s) are not in the WAITING or ACTIVE SCHEDULE, the MCP notifies the operator that an invalid request has been entered.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number1

This field must contain a mix number of a program currently in the WAITING or ACTIVE SCHEDULE and specifies the program to be removed from the WAITING or ACTIVE SCHEDULE.

mix-number2

This field must contain a mix number of a program currently in the WAITING or ACTIVE SCHEDULE and is used to specify the lower limit of a range of mix numbers. <mix-number2> must have a value less than <mix-number3>.

mix-number3

This field must contain a mix number of a program currently in the WAITING or ACTIVE SCHEDULE and is used to specify the upper limit of a range of mix numbers. <mix-number3> must have a value greater than <mix-number2>.

=/=

The =/= key symbol causes all the programs in the WAITING and ACTIVE SCHEDULE to be removed.

Example:

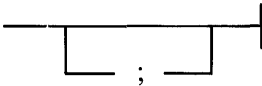
Command	Response
RS 33, 34, 35, 36;	#33 RS-ED #34 RS-ED #35 RS-ED #36 NOT SCHEDULED – NOT RS-ED
RS =/=;	#24 RS-ED #24 RS-ED
RS 1246 - 1248	#1246 RS-ED #1247 RS-ED #1248 RS-ED

RT

RT (Remove Multipack File Table Entry)

The RT system command removes an entry from the multipack file table.

Remote ODT syntax:

— RT <file-identifier> — 

Semantics:

file-identifier

This field, which must contain a valid multipack file identifier, specifies the name of the file entry to be removed from the multipack file table.

Examples:

RT USER/A/B;

RT BASEPACK/MASTER/;

RY (Ready Peripheral)

The RY system command readies a peripheral unit and makes it available to the MCP.

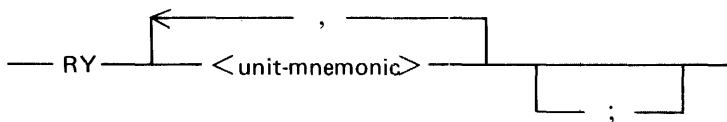
If the designated unit is not in use and is in the remote status, the RY system command causes all exception flags maintained by the MCP for the specified unit to be reset. After the unit has been made ready, the MCP attempts to read the file label (input devices only).

If the designated unit is a disk pack or cartridge, the RY system command may invoke the SYSTEM/PANDA program from the name table to verify disk structures, absolutize the pack, remove temporary files, and correct end of file (EOF) pointers for protected files.

A RY system command referencing a card reader device causes the MCP to read the next card in the input hopper. If this card is a control card, the MCP does what is requested.

The only exception to this is when a card reader has been reserved by a program by mix number because the COMPILE or EXECUTE program control instruction immediately precedes the DATA program control instruction without an intervening END program control instruction. Without operator intervention (IL or UL system command), such reserved card files can be opened only by the program for which they are reserved and, if unopened, remain reserved even after the program terminates or the operator enters the RS system command for the program. The RY system command has no effect on a card reader that is reserved in this manner. The CL system command permits the operator to remove the reserved condition from the card reader and allow it to be assigned to any program that opens it with the correct file identifier. To completely free the card reader by removing the file identifier, follow the CL system command with the RY system command.

Remote ODT syntax:



Semantics:

unit-mnemonic

This field, which must be a valid unit mnemonic, specifies the device to ready.

Examples:

Command	Response
RY DPB;	DPB LABELED MASTER (U) #884071
RY MTC;	
RY LPA, LPB;	

SB (Seconds Before decay)

The SB system command sets or interrogates the seconds-before-decay attribute for code segments for the task specified.

The SB system command is valid only if the MPRI MCP option is set. The value specified is in units of tenths of a second. The default value is 0 and the range is from 0 to 1220 inclusive.

If <integer> is omitted, the current setting of seconds-before-decay assigned to the program is displayed.

Refer to appendix A for further information regarding Decay Intervals.

Remote ODT syntax:

```

— <mix-number> SB — [ ] = [ ] <integer> [ ] ; [ ]

```

Semantics:

mix-number

This field, which must contain a mix number, specifies the program in which to interrogate or change the value of the seconds-before-decay attribute.

integer

This field, which must contain an integer in the range 0 to 1220 inclusive, specifies the amount of time (in tenths of a second) for a code segment to remain in memory before it is decayed to the next memory priority.

Examples:

Command	Response
1243SB = 31;	NDL/HANDLER =1243 SECONDS BEFORE DECAY CHANGED TO 31
1244SB =;	CANDE =1244 SECONDS BEFORE DECAY = 10

SD (Assign Additional System Drives)

The SD system command assigns additional SYSTEM drives to the MCP.

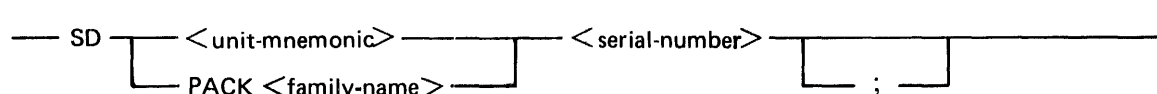
The SD system command causes the specified disk to be purged after verifying that <serial-number> is correct and adds the purged disk to the SYSTEM disks already on the system.

At coldstart time, there is only one SYSTEM disk; additional disk drives can be added by the SD system command. Once a SYSTEM drive has been added to the system, it cannot be removed without performing a coldstart operation.

The following message is displayed when the new SYSTEM disk is linked to the system.

<unit-mnemonic> IS NOW A SYSTEM PACK

Remote ODT syntax:



Semantics:

serial-number

This field must contain a 6-digit number and is the serial number of the user disk to be made into a SYSTEM disk.

unit-mnemonic

This field, which must contain a valid user-disk specifies the disk drive that is to be made into a SYSTEM disk drive.

PACK <family-name>

The PACK keyword specifies that a disk identifier follows instead of a unit mnemonic.

Example:

SD DPB 102182

SE (Switch Enable)

The SE system command enables the sensing of the 24 Data Entry Switches on the B 1000 console (while in RUN mode) by specific classes of programs and firmware.

Only SDL and MIL programs contain the source language constructs required to sense the Data Entry Switches. In both languages, the CONSOLE_SWITCHES construct is used, and provides the 24-bit image for the Data Entry Switches.

Remote ODT syntax:

```
SE <integer> ;
```

Semantics:

integer

This field must contain an integer in the range 0 to 15 inclusive.

The value of <integer> specifies the programs that are allowed to sense the Data Entry Switches according to the following table.

Value of <integer>	Program
0	Disable Switches
2	Normal-state Programs
4	Interpreters
8	GISMO

<integer> can also be any sum of the above values, in which case multiple classes of programs are allowed to sense the Data Entry Switches.

Examples:

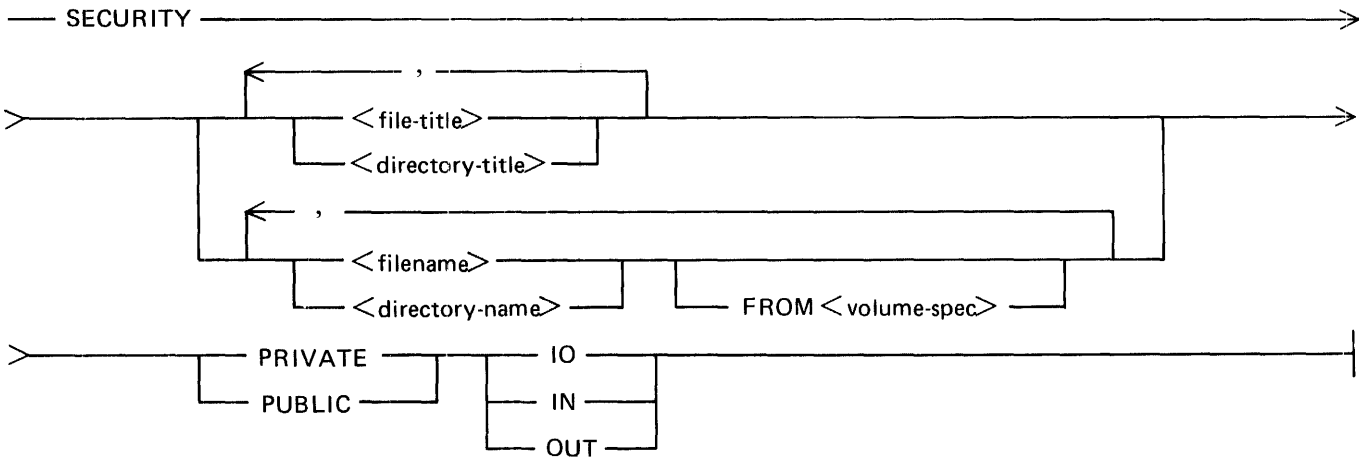
```
SE 2;
```

```
SE 10;
```

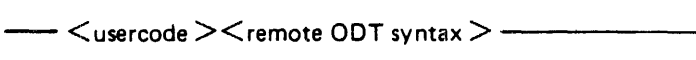
SECURITY

The SECURITY system command changes the security attributes of files on disk.

Remote ODT syntax:



Workstation syntax:



Semantics:

IO

The IO keysymbol allows the file to be opened for input and output.

IN

The IN keyword restricts the file to be opened for input only.

OUT

The OUT keyword restricts the file to be opened for output only.

PRIVATE

The PRIVATE keyword restricts access to the file to 1) usercodes that are privileged (*PRIV) usercodes or 2) usercodes that have the same name as the first part of the file title.

PUBLIC

The PUBLIC keyword allows access to the file by any user.

Example 1:

```
SECURITY AB/XY PRIVATE IO;
```

Changes the security of file AB/XY on DISK to private for input and output.

Example 2:

```
SECURITY Z ON PACK PUBLIC IN;
```

Changes the security of file Z on the disk PACK to public for input only.

SL (Set LOG)

The SL system command sets the LOG or ODTL MCP options, and allocates the disk area required.

The MCP responds with one of the following messages when an SL system command is entered.

LOG [ODTLOG] NOW SET NOW SET **CLEAR START REQUIRED**
~~FOR ACCURATE LOG INFORMATION~~

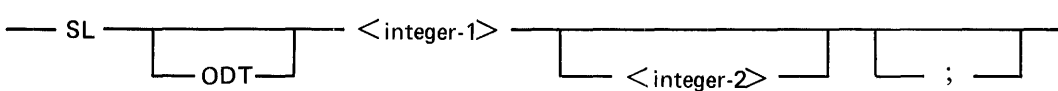
X12466.

If there is not sufficient disk space for the first area of the log file, the following message is displayed.

NO SPACE TO BUILD LOG [ODTLOG]

If the value of <integer-1> is zero, the LOG or ODTL system option is reset and the log file is transferred (as though a TL system command had been entered). A new SYSTEM/LOG or SYSTEM/ODTLOG file is not created.

Remote ODT syntax:



Semantics:

integer-1

This field, which must contain an integer in the range 100 to 10000 inclusive, specifies the size of each disk area to be assigned to the LOG or ODTLOG file.

integer-2

This field, which must contain an integer in the range 2 to 105 inclusive, specifies the maximum number of disk areas desired.

Examples:

SL 1000;

SL 250 5;

SL 0;

SL ODT 100 105;

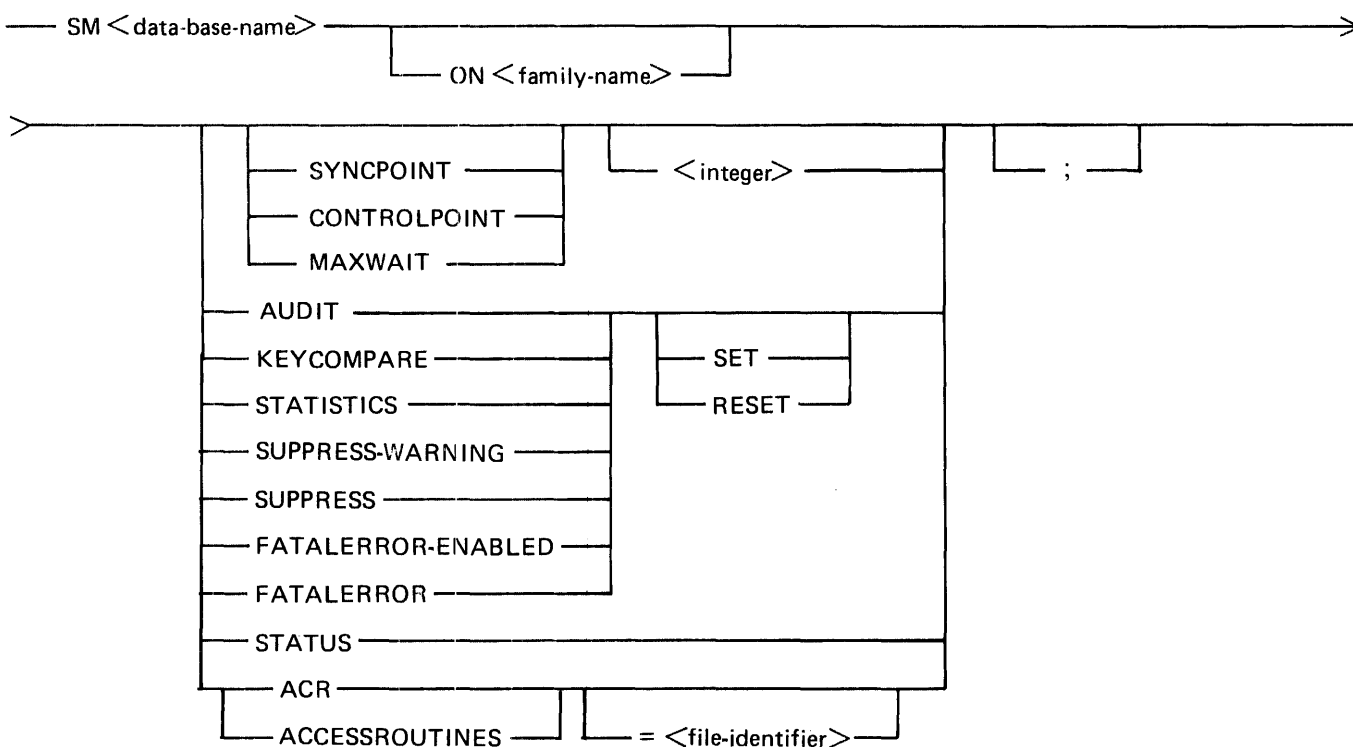
SM (Set Database Parameters)

The SM system command is used in conjunction with the B 1000 DMSII system and interrogates or dynamically sets certain parameters of a DMSII data base. The parameters can also be specified in the DASDL source. Refer to the DASDL manual for more information.

If an integer is omitted when specifying the SYNCPOINT, CONTROLPOINT, or MAXWAIT parameters, or if SET or RESET is omitted when specifying the AUDIT or KEYCOMPARE parameters, or if a file-identifier is omitted when specifying the accessroutines parameter, the current setting of the parameter is displayed by the MCP. The data base need not be inactive in order to interrogate the parameters.

If the data base is active, the MCP does not allow any of the parameters specified by the SM system command to be changed. If the data base does not have the AUDIT option specified in the DMS/DASDL compiler source file, the AUDIT, CONTROLPOINT, and SYNCPOINT parameters cannot be changed by the SM system command.

Remote ODT syntax:



Semantics:

integer

This field, which must contain an integer, specifies the value to use for the CONTROLPOINT, SYNCPOINT, and MAXWAIT keywords.

ACCESSROUTINES or ACR

The ACCESSROUTINES keyword causes the name of the DMSII access routine used by the data base to be displayed. If ACCESSROUTINES is followed by a <file-identifier> equation, the name of the access routine is changed. The default name for the access routine file is DMS/ACR, but may be changed for test purposes.

AUDIT

The AUDIT keyword changes or interrogates the AUDIT parameter in the specified data base. If the SET or RESET keywords follow the AUDIT keyword, the AUDIT parameter is set or reset, respectively. The AUDIT option must be specified in the DASDL source file in order for the AUDIT keyword in the SM system command to be valid. If AUDIT is reset on a data base then any program using begin or end transaction commands will be DSed.

CONTROLPOINT

The CONTROLPOINT keyword assigns or interrogates the CONTROL POINT parameter in the specified data base. If <integer> follows the CONTROLPOINT keyword in the SM system command, the CONTROL POINT parameter is assigned the value of <integer> syncpoints per controlpoint.

FATALERROR-ENABLED or FATALERROR

The FATALERROR-ENABLED or FATALERROR keywords query, set, or reset the DM_FATALERROR_ENABLED flag in the DMS globals. This flag controls whether a job that is discontinued, (DS or DP system command), generates a FATALERROR exception if it cannot obtain a system lock to complete its operation. If this flag is reset (by default) then the job is not discontinued. If the flag is set, then a discontinuing of the job causes a FATALERROR exception, closing the data base for all programs.

KEYCOMPARE

The KEYCOMPARE keyword sets, resets, or interrogates the KEY COMPARE parameter in the specified data base. If the SET or RESET keyword follows the KEYCOMPARE keyword in the SM system command, the KEY COMPARE parameter is set or reset, respectively.

MAXWAIT

The MAXWAIT keyword assigns or interrogates the MAXWAIT parameter in the specified data base. If <integer> follows the MAXWAIT keyword in the SM system command, the MAXWAIT parameter is assigned the value of <integer>.

ON

The ON keyword is required if <family-name> is to follow in the syntax of the SM system command and specifies that the data base dictionary resides on a user disk.

RESET

The RESET keyword is valid for the AUDIT, KEYCOMPARE and STATISTICS keywords, and resets the AUDIT or KEY COMPARE parameters in the specified data base.

SET

The SET keyword is valid for the AUDIT and KEYCOMPARE keywords and sets the AUDIT or KEY COMPARE parameters in the specified data base.

STATISTICS

The STATISTICS keyword causes DMSII statistics to be displayed when the data base is closed by any program. The following statistics are displayed:

- Number of transactions
- Number of exception (including NOTFOUND exception) conditions
- Processor time for DMSII access routines
- Number of update operations
- Number of non-update operations

STATUS

The STATUS keyword allows rapid interrogation of the abnormal data base conditions. If the condition is not abnormal, then the status NORMAL is reported. If the condition is abnormal, then the status reported is one or more of the following:

RECOVERY REQUIRED indicates that the data base was not closed normally at the end of its last update run because a clear/start or fatal error occurred.

WRITE ERROR indicates that an irrecoverable I/O error occurred on the dictionary.

RECOVERY IN PROGRESS indicates that DMS/RECOVERDB has been initiated, but has not completed executing. DMS/RECOVERDB may be currently running or may have been DS'ed.

REORGANIZATION indicates that DMS/REORGANIZATION has been initiated, but has not completed executing. DMS/REORGANIZATION may be running currently or may have been DS'ed.

STRUCTURE WRITE ERROR indicates that one or more of the structures had a write error. DMS/DBMAP should be executed to identify the structure.

SUPPRESS-WARNINGS or SUPPRESS

The SUPPRESS-WARNINGS or keywords query, set or reset the DM__SUPPRESS__WARNINGS flag in the DMS globals. This flag controls the display of the message: DMS ACCESSROUTINES HAVE BEEN HUNG FOR 30 SECONDS WAITING FOR A SYSTEM LOCK.

SYNCPOINT

The SYNCPOINT keyword changes or interrogates the SYNCPOINT parameter in the specified data base. If <integer> follows the SYNCPOINT keyword, the SYNCPOINT parameter is assigned the value of <integer> transactions per syncpoint.

data-base-name

This field, which must contain a valid data base identifier, specifies the data base for which to interrogate or change the specified parameters.

family-name

This field, which must contain a valid disk identifier, specifies the user disk on which the data base dictionary resides.

Examples:

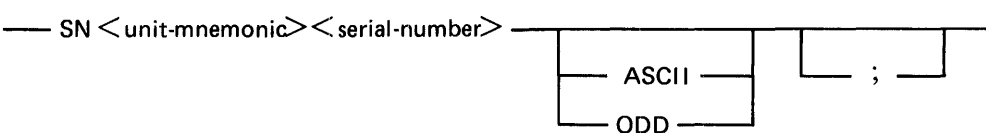
Command	Response
SM STUDENTDB CONTROLPOINT 25;	CONTROLPOINT CHANGED FROM 10 TO 25
SM PAYROLL ON USER1 SYNCPOINT;	SYNCPOINT = 5
SM FTRDB AUDIT;	AUDIT IS SET
SM INVENTORY ACCESSROUTINES;	ACCESSROUTINES = DMS/ACR FOR DATABASE INVENTORY

SN (Assign a Tape Serial Number)

The SN system command initializes and purges a magnetic tape, assigning a volume serial number to the tape label. An ANSI label is written to the magnetic tape. Any tape that does not contain a valid ANSI label cannot be purged with the PG system command. This includes both unlabeled tapes and those that have the Burroughs standard label.

ANSI tape labels can be maintained in either EBCDIC or 8-bit ASCII recording mode. The label recording mode is determined by the SN system command, and remains unaltered (even if the PG system command is used) until explicitly changed by another SNL or SN system command. The recording mode of the tape label is completely independent of the mode in which the data is recorded on the tape.

Remote ODT syntax:



Semantics:

unit-mnemonic

This field, which must contain a valid magnetic tape unit mnemonic, specifies the tape drive on which to perform the SN.

serial-number

This field, which must contain a 6-character alphanumeric string, specifies the serial number of the tape label, and remains in all labels on the tape even if the PG system command is used. <serial-number> can be explicitly changed by another SN or SNL system command.

ASCII

The ASCII keyword causes the recording mode to be EVEN and is only valid for 7-track tape drives.

ODD

The ODD keyword causes the recording mode to be ODD and is only valid for 7-track tape drives.

Examples:

SN MTA 123456;

SN MTC BACKUP ASCII;

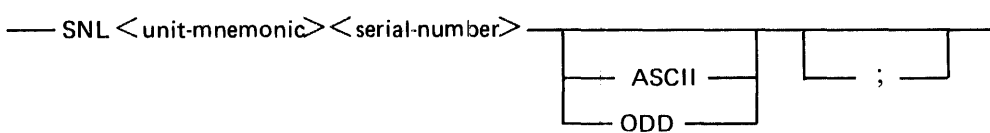
SN MTD 012681 ODD;

SNL (Assign a Tape Serial Number and Lock)

The SNL system command initializes and purges a magnetic tape, assigning a volume serial number to the tape label. An ANSI label is written to the magnetic tape. Any tape that does not contain a valid ANSI label cannot be purged with the PG system command. This includes both unlabeled tapes and those that have the Burroughs standard label. The tape is locked following initialization if the SNL system command is used, and is readied following initialization if the RY system command is used.

ANSI tape labels can be maintained in either EBCDIC or 8-bit ASCII recording mode. The label recording mode is determined by the SNL system command, and remains unaltered (even if the PG system command is used) until explicitly changed by another SNL or SN system command. The recording mode of the tape label is completely independent of the mode in which the data is recorded on the tape.

Remote ODT syntax:



Semantics:

unit-mnemonic

This field, which must contain a valid magnetic tape unit mnemonic, specifies the tape drive on which to perform the SNL.

serial-number

This field, which must contain a 6-character alphanumeric string, specifies the serial number of the tape label, and remains in all labels on the tape even if the PG system command is used. <serial-number> can be explicitly changed by another SN or SNL system command.

ASCII

The ASCII keyword causes the recording mode to be EVEN and is only valid for 7-track tape drives.

ODD

The ODD keyword causes the recording mode to be ODD and is only valid for 7-track tape drives.

Examples:

```
SNL MTA 123456;
```

```
SNL MTC BACKUP ASCII;
```

```
SNL MTD 012681 ODD;
```

SO (Set Option)

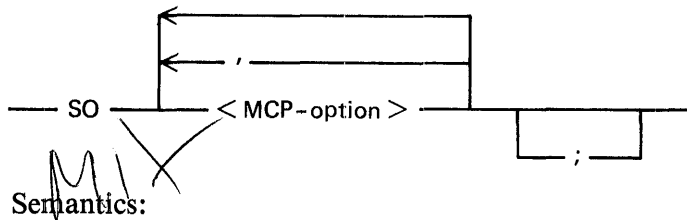
The SO system command sets MCP options. Refer to section 3 for a complete list of the MCP options that can be specified with the SO system command.

The MCP replies with verification that the option has been set after each SO system command.

The LOG and ODTL cannot be set with the SO system command. If an attempt is made to set these options, the MCP displays a message indicating that the LOG or ODTL options are locked.

The TO system command can be used to determine which options are currently set. The option indicator equals 1 when set and 0 when reset.

Remote ODT syntax:



Semantics:

MCP-option

This field can contain any valid MCP option and specifies the option to be set.

Examples:

Command	Response
SO COPY;	COPY ALREADY SET
SO LIB, RMOV;	LIB = 1 RMOV = 1
SO LOG;	LOG LOCKED

SP (Change Schedule Priority)

The SP system command changes the schedule priority of a program currently in the active or waiting schedule.

The schedule priority is separate from the priority of the program when it is in the mix and specifies the order in which programs are allowed to enter the mix. A program with a given schedule priority enters the mix prior to any program in the active schedule with a lower priority.

Remote ODT syntax:

— SP <mix-number> ———— <integer> ————
 — <mix-number> SP ———— ; ————

Workstation syntax:

— <usercode><remote ODT syntax> ————

Semantics:

integer

This field, which must contain an integer in the range 0 to 14 inclusive, specifies the value for the schedule priority.

mix-number

This field must contain a mix number of a program in the active or waiting schedule and specifies the program to which the new schedule priority is to be assigned.

Examples:

Command	Response
1974 SP 14;	#1974 SCHEDULE PRIORITY CHANGED TO 14.
SP 1700 10;	#1700 SCHEDULE PRIORITY CHANGED TO 10.

SQ (Squash Disk)

The SQ system command initiates the SYSTEM/SQUASH program to perform a disk squash.

If the files on a disk pack are volatile, then squashing the disk should be performed regularly. This prevents the buildup of a large number of small disk areas, also known as checkerboarding, that can reduce the amount of disk space available to the system.

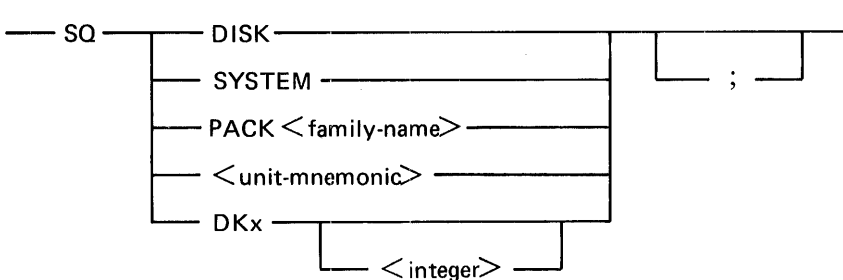
When squashing a disk, the MCP moves areas of data to a numerically-lower disk address in order to alleviate checkerboarding.

Either the SYSTEM disk or a user disk can be squashed. Each SQ system command affects only one disk of a multiple SYSTEM disk configuration or one EU of a head-per-track device. The DISK or SYSTEM keywords can be used to specify a SYSTEM disk squash.

Before invoking the SYSTEM/SQUASH program on a system disk, the MCP stops all tasks except the SYSTEM/ODT program, and for systems without an ODT, the handler. The tasks are restarted when the SYSTEM/SQUASH program terminates. During a squash of the SYSTEM disk, input to the ODT is restricted to commands for the SYSTEM/SQUASH program; all other input is rejected with an appropriate message.

Refer to the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the SYSTEM/SQUASH program.

Remote ODT syntax:



Semantics:

DISK

The DISK keyword indicates that the SYSTEM disk is to be squashed.

SYSTEM

The SYSTEM keyword indicates that the SYSTEM disk is to be squashed.

PACK

The PACK keyword indicates that a user disk is to be squashed.

family-name

This field, which must contain a disk identifier, specifies the name of the disk to be squashed.

unit-mnemonic

This field can contain any of the disk unit mnemonics. The disk unit mnemonics are DCx and DPx.

DKx

The DKx key symbol specifies a head-per-track SYSTEM disk on which the squash operation is to be performed, where the lower-case letter x is the disk drive of the SYSTEM disk.

integer

This field, which must contain an integer, specifies the electronics unit (EU) of a head-per-track SYSTEM disk.

Examples:

SQ SYSTEM

SQ DPA

SQ PACK TESTPACK

ST (Suspend Processing)

The ST system command temporarily suspends the processing of a program in the mix.

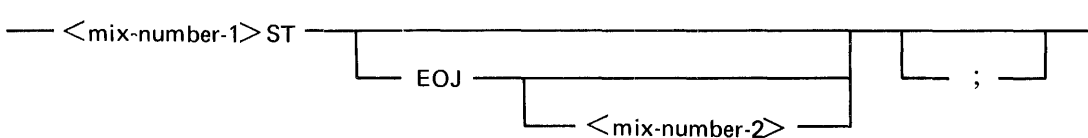
The MCP does not suspend the program until all input/output (I/O) operations in progress are complete.

A suspended program retains the mix number and peripherals assigned to it; the MCP uses this information to identify the program when referenced by another system command.

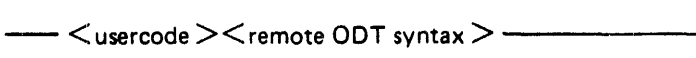
To restart a program after it has been suspended, enter the OK or GO system command. If for any reason the conditions necessary for the program to run are not met when the OK or GO system command is entered, the MCP does not restart the program.

By using the EOJ key symbol, a program that is suspended can also be restarted as a result of the termination of another program.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number-1

This field, which must contain a mix number, specifies the program to be temporarily suspended.

EOJ

The EOJ keyword causes the specified program to be temporarily suspended until another program in the mix reaches end of job (EOJ). The EOJ option in the ST system command is useful when thrashing has been detected in order to make memory space available for other running programs. Refer to appendix A for a complete description of thrashing detection.

mix-number-2

This field, which must contain a mix number of a currently executing program, specifies the task that must terminate in order for the suspended task to be restarted.

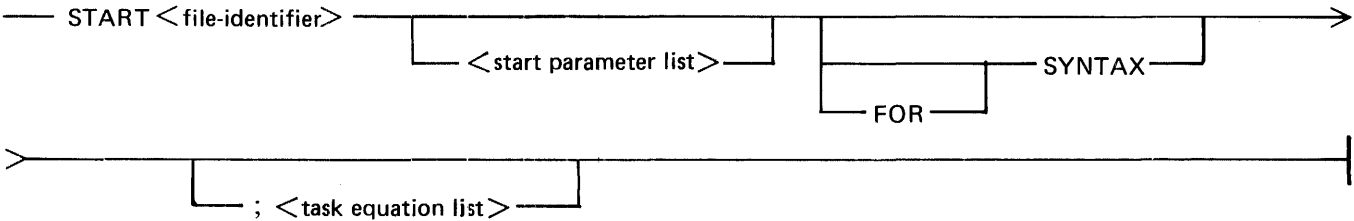
Examples:

Command	Response
823 ST;	DMPALL = 823 STOPPED. TIME = 14:23:22.1
947 ST EOJ;	CANDE = 947 STOPPED UNTIL AN EOJ. TIME = 07:03:45.6
1234 ST EOJ 1201;	DMPALL = 1234 STOPPED UNTIL EOJ OF JOB #1201. TIME = 15:11:01.0

START

The START system command requests that the WFL job be initiated. The START command requires that the command end with the end of text, that is, another command cannot follow the START command. Refer to the B 1000 Systems Work Flow Language (WFL) Language Manual for a complete description of the WFL system.

Syntax:



<file-title>

This field must refer to a file containing the source of a WFL <job> to be initiated.

<start parameter list>

This field is a boolean, integer, or string expression that is substituted for the parameter identifiers in the <job parameter list> of the STARTed job.

SYNTAX

The SYNTAX keyword indicates that the started job is compiled for syntax only and execution does not occur.

<task equation list>

This field specifies task attributes for the initiated job.

SV (Save Peripheral Unit)

The SV system command makes a peripheral device inaccessible to all tasks until a clear/start operation occurs, or until a RY or CL system command is used to ready the device.

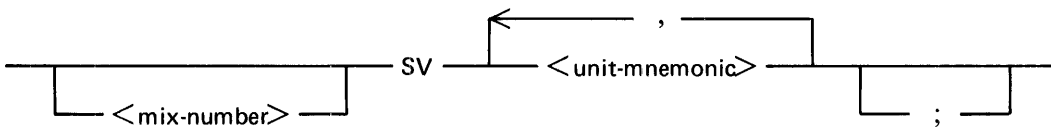
If the device is in use, the MCP displays the following message.

<unit-mnemonic> TO BE SAVED

The device is made available when the task that the device was saved for closes the file that was using the device.

Disk devices (including DISKETTE) cannot be specified with the SV system command.

Remote ODT syntax:



Semantics:

mix-number

This field must contain a mix number and causes the device to be saved for the program specified. No other programs can access the device until the program specified relinquishes the device.

unit-mnemonic

This field, which must contain a non-disk unit mnemonic, specifies the device to be saved.

Examples:

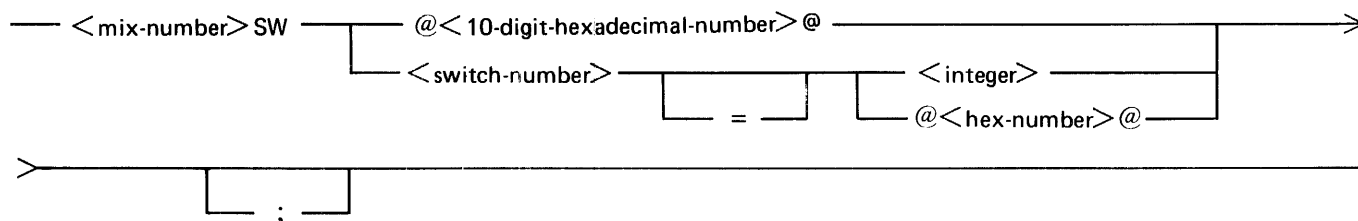
Command	Response
SV MTA;	MTA SAVED
1456 SV LPA;	LPA SAVED FOR MIX #1456
SV MTB, LPA;	MTA SAVED LPA SAVED

SW (Set Switch)

The SW system command sets programmatic switches for the specified program.

Programmatic switches can also be set using the SWITCH program control instruction. Refer to section 4 for a complete description of the SWITCH program control instruction.

Remote ODT syntax:



Workstation syntax:



Semantics:

mix-number

This field, which must contain a mix number of an executing program, specifies the program for which to set the programmatic switch(es).

10-digit-hexadecimal-number

This field, which must contain a hexadecimal number in the range 0000000000 to FFFFFFFF inclusive, specifies the values of program switches 0 through 9, respectively.

switch-number

This field, which must contain an integer in the range 0 to 9 inclusive, specifies the program switch to set.

integer

This field, which must contain a 1-digit integer in the range 0 to 9 inclusive, specifies the value of <switch-number>.

hex-number

This field, which must contain a hexadecimal number in the range 0 to F inclusive, specifies the value at which to set <switch-number>.

Examples:

25 SW 1 = @F@;

2 SW 8 = 6;

1473 SW = @0123456789@;

SY (System Program Pack)

The SY system command interrogates or changes the disk device on which the MCP expects to find certain utility programs. This disk device is called the system program pack. The utility programs are those that are normally executed by a system command instead of the EXECUTE program control instruction (for example, the SYSTEM/BACKUP program is normally executed with the PB system command). The SY system command allows these utility programs to be moved from the system disk while allowing the operator to use the convenient system commands for their execution. It is intended for sites having extremely limited system disk capacity.

If no parameter is included with the SY system command, the current system program pack is displayed.

The system program pack defaults to the system disk at the time of a coldstart operation. If the system program pack is changed to a user pack, the system disk can later be indicated by specifying a null quoted string (" ") as a parameter.

Table 5-1 lists the utility programs that can be executed through a system command and indicates the pack on which the MCP expects to find the program. The utility programs that are expected to be located on the system disk only cannot be affected by the SY system command. The remaining utility programs are expected to be located on the system program pack.

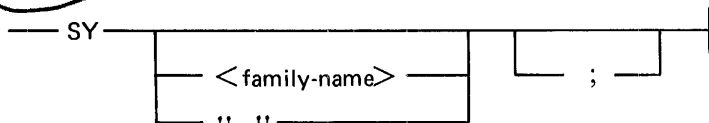
Table 5-1. Utility Programs and their Associated System Command

Utility Program	System Command	Expected Location
BNA/NSM	NW	system disk
DISKETTE/COPY	OL FDx	system program pack
DMS/RECOVERDB	RC	system disk
SYSTEM/BACKUP	PB	system disk
SYSTEM/COPY	various	system disk
SYSTEM/ELOGOUT	ET	system program pack
SYSTEM/IDA	PM	system program pack
SYSTEM/LDCONTRL	LD	system program pack
SYSTEM/LOGOUT	LG/LN	system program pack
SYSTEM/ODTLOGOUT	LG ODT	system program pack
SYSTEM/PANDA	various	system disk
SYSTEM/SQUASH	SQ	system disk
SYSTEM/WFL	various WFL	system disk

Remote ODT syntax:

only from ODT

?



Semantics:

family-name

This field can contain any disk identifier and specifies the name of the disk that is to be the system program pack.

" "

The " " key symbol indicates that the system disk is to be the system program pack.

Examples:

Command	Response
SY	SYSTEM PROGRAM PACK= SYSTEM DISK
SY USERPACK	SYSTEM PROGRAM PACK= "USERPACK".
SY " "	SYSTEM PROGRAM PACK= SYSTEM DISK

TD (Time and Date)

The TD system command causes the MCP to display the current system time and date. The following is the format of the response generated by the MCP.

TIME = <hours>:<minutes>:<seconds>.<tenths-of-second>
DATE = <month>/<day>/<year> <name-of-day>
(JULIAN DATE = <year><julian-day>)

The time displayed is based on a 24-hour clock. For example, 08:10:25.3 AM is displayed as 08:10:25.3, and 08:10:25.3 PM is displayed as 20:10:25.3.

Syntax:

— TD — [] ; []

Example:

Command	Response
TD;	TIME = 15:30:54.3 DATE = 01/26/81 MONDAY (JULIAN DATE = 81026)

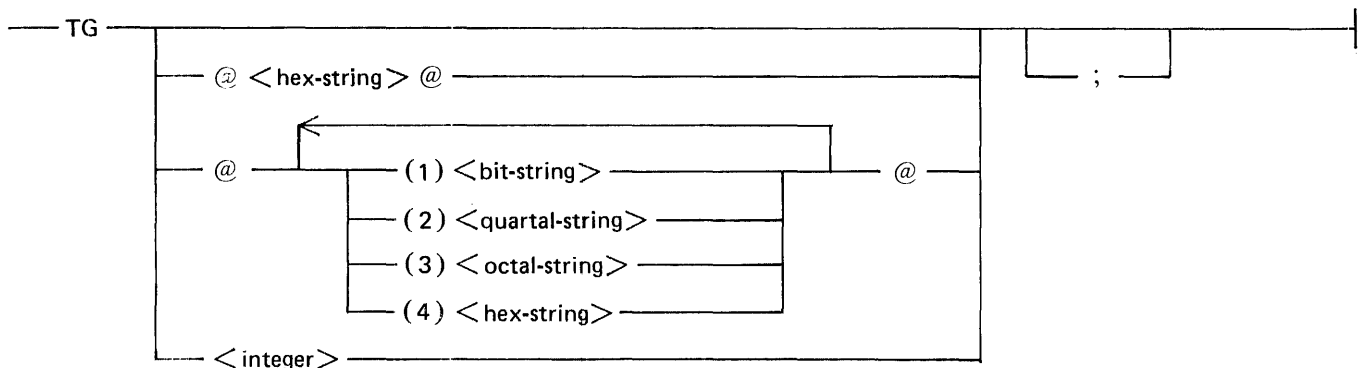
TG (Trace Gismo)

The TG system command interrogates or changes the group of flags that instruct the GISMO program to trace various system events. If no parameter is specified, the TG system command interrogates and displays the current settings of the trace flags. If a parameter is specified, the TG system command sets the flags according to the value of the parameter. The memory required for the GISMO trace code and trace table space is allocated at clear/start time if the TG flags are non-zero at that time. Once the memory is allocated, a clear/start operation is not required to invoke any change. The DEBUG system option must be set for any tracing. The MICRO-MCP/DEBUG program is required for tracing MMCP activity.

The parameter maps onto a 24-bit field used by the GISMO program to determine which functions to place in the trace table. If a bit is set, the corresponding function is traced; otherwise, the function is not traced. The 24-bit field is mapped as follows.

Bits	Function
0	Trace channel 0 activity.
1	Trace channel 1 activity.
.	.
.	.
.	.
14	Trace channel 14 activity.
15	Trace all GISMO functions except interrupt handling, scheduler functions, and the timer.
16	Trace port activity (multiline control and master/slave communication on dual-processor systems).
17	Not used.
18	Trace MMCP activity (MICRO-MCP/DEBUG must be used).
19	Trace SMCP CONDITIONAL.HALTS.
20	Trace interrupt handling, scheduler functions, and the timer.
21	Time stamp all trace entries
22	Trace data transfers for those channels specified by bits 0 through 14.
23	Reserved for GISMO use.

Remote ODT syntax:



Semantics:

hex-string

This field can contain any hexadecimal number and maps onto the 24-bit trace field.

bit-string

This field can contain any bit string of zeros (0) and ones (1) and maps onto the 24-bit trace field.

quartal-string

This field can contain any quartal string and maps onto the 24-bit trace field.

octal-string

This field can contain any octal string and maps onto the 24-bit trace field.

(1), (2), (3), (4)

The (1), (2), (3), and (4) keysymbols specify that a bit, quartal, octal, or hexadecimal string immediately follows.

integer

This field can contain any integer in the range 0 to 16,777,215 inclusive and maps onto the 24-bit trace field.

Examples:

TG

```
GISMO TRACE FLAGS = @000204@ CHANNEL MASK @0002@ TIMESTAMP
```

TG @001204@

```
GISMO TRACE FLAGS CHANGED FROM @000204@ TO @001204@ CHANNEL MASK  
@0012@ TIMESTAMP
```

The second example sets bits 11, 14, AND 21 of the trace field. This causes tracing of I/O activity for channels 11 and 14 with a time stamp for each trace entry.

TI (Time Interrogation)

The TI system command interrogates the amount of processor time the specified program has accumulated at the time the interrogation is made.

The following is the format of the response displayed by the MCP.

```
USER INTERP = hh:mm:ss.t; MMCP = hh:mm:ss.t; SMCP = hh:mm:ss.t  
[DMS INTERP = hh:mm:ss.t; MMCP = hh:mm:ss.t; SMCP = hh:mm:ss.t]  
[IBASIC INTERP = hh:mm:ss.t; MMCP = hh:mm:ss.t; SMCP = hh:mm:ss.t]  
TOTAL INTERP = hh:mm:ss.t; MMCP = hh:mm:ss.t; SMCP = hh:mm:ss.t  
TOTAL PROCESSOR = hh:mm:ss.t; ELAPSED = hh:mm:ss.t
```

Syntax:

— <mix-number> TI _____
 └───┬───┘
 ; ───┘

Semantics:

mix-number

This field, which must contain a mix number, specifies the program for which to interrogate the accumulated processor time.

Example:

Command	Response
84 TI;	DMS/INQUIRY = 84 USER INTERP = 3:15.0; MMCP = 23.1; SMCP = 12.4
	DMS/INQUIRY = 84 DMS INTERP = 1:04.4; MMCP = 0; SMCP = 18.9
	DMS/INQUIRY = 84 TOTAL INTERP = 4:19.4; MMCP = 23.1; SMCP = 31.3
	DMS/INQUIRY = 84 TOTAL PROCESSOR = 5:13.8; ELAPSED = 23:54.6

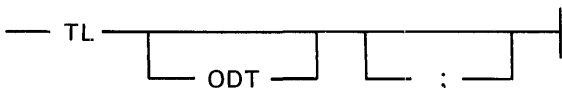
TL (Transfer LOG)

The TL system command transfers the information in the SYSTEM/LOG or SYSTEM/ODTLOG files to LOG/#<integer> or ODTLOG/<date-and-time>, respectively, where <date-and-time> has the form <MMDDYYHHMM>. To transfer and print the log files, refer to the LG system command.

If the ODT keyword is omitted in the TL system command, the SYSTEM/LOG file is transferred. If the ODT keyword is included, the SYSTEM/ODTLOG file is transferred.

The TL system command creates a new (empty) log file.

Remote ODT syntax:



Semantics:

ODT

The ODT keyword causes the SYSTEM/ODTLOG file to be transferred. If the ODT keyword is omitted, the SYSTEM/LOG file is transferred.

Example:

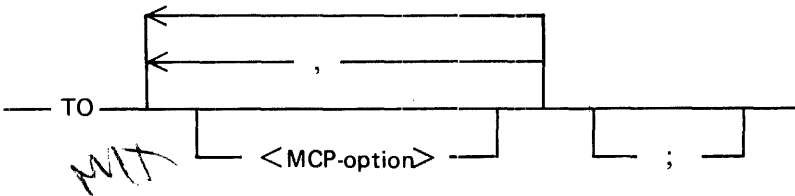
Command	Response
TL;	LOG TRANSFERRED TO LOG/#000025
TL ODT;	SYSTEM/ODTLOG CHANGED TO ODTLOG/0308830844

TO (Display Options)

The TO system command interrogates the status of MCP options. Refer to section 3 for a complete description of the MCP options.

If <MCP-option> is omitted in the TO system command, all the MCP options and associated settings are displayed. If the MCP option has a value of 1, the MCP option is set. If the MCP option has a value of 0, the MCP option is reset.

Remote ODT syntax:



Semantics:

MCP-option

This field can contain any valid MCP option name and specifies the option to be interrogated. If <MCP-option> is not specified, all the MCP options and their associated settings are displayed.

Examples:

Command	Response
TO LOG;	LOG =1
TO BOJ, DATE;	BOJ =1 DATE=0
TO;	

```
AMCS=0 BOJ =1 BREL=1 CHRG=0 CLOS=0 COPY=0 DATE=1 DBUG=0 DISP=1
DUMP=1 EOJ =1 FLMP=1 LAB =1 LIB =1 LOG =0 MEM =1 MPRI=1 ODTL=1
OPEN=0 PBD =1 PBT =0 RMOV=1 RMSG=0 SCHM=0 SQRM=0 TERM=0 THR =1
TIME=1 TRMD=1 VLCP=1 VLIO=1 WFL =0 ZIP =0
```

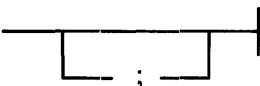
TR (Time Change)

The TR system command changes the current value of the time maintained by the MCP.

The time specified is designated according to a 24-hour clock, and must be four numeric digits in length.

The MCP displays the new time after the TR system command is entered.

Remote ODT syntax:

— TR <integer> — 

Semantics:

integer

This field, which must contain a 4-digit integer in the range 0000 to 2400 inclusive, specifies the time according to a 24-hour clock.

Example:

Command	Response
TR 1919;	TIME = 19:19:00.0

TS (Test Switches)

The TS system command interrogates the switches set by the SW system command or by the SWITCH program control instruction.

The output of the TS system command is in hexadecimal format.

Syntax:

— <mix-number> TS ————
 └──┬──┘
 ; —┘

Semantics:

mix-number

This field, which must contain a mix number of an executing program, specifies the program for which to interrogate the settings of the program switches.

Example:

Command	Response
695 TS;	CANDE =695 SWITCHES = @0000010000@

UL

UL (Assign Unlabeled File)

The UL system command designates the device on which a particular unlabeled input file is located. This instruction is entered in response to the following MCP message:

FILE NOT PRESENT

The UL system command is used only if the specified device is to be acted on as an unlabeled file. The MCP assumes that the file on the designated unit is the file requested by the program that caused the FILE NOT PRESENT message. The device specified in the UL system command must be ready when the instruction is entered.

Unlabeled card reader files are not allowed by the MCP. Therefore, the UL system command with a card reader device functions the same as the IL system command.

Remote ODT syntax:

— <mix-number> UL <unit-mnemonic> _____ |
 | |
 | <integer> | |
 | |
 | |

Workstation syntax:

— <usercode> <remote ODT syntax> _____ |

Semantics:

mix-number

This field, which must contain a mix number, specifies the program that is requesting the unlabeled file.

unit-mnemonic

This field, which must contain a valid unit mnemonic of an input device, specifies the device on which the unlabeled file is located.

integer

This field, which must contain an integer, specifies the number of blocks for the MCP to space forward prior to reading the first data block requested by the program.

Examples:

The following UL system command causes the unlabeled input tape file on tape unit MTA to be assigned to the program whose mix number is 7404.

7404 UL MTA;

The following UL system command causes the first three blocks on tape unit MTA to be skipped.

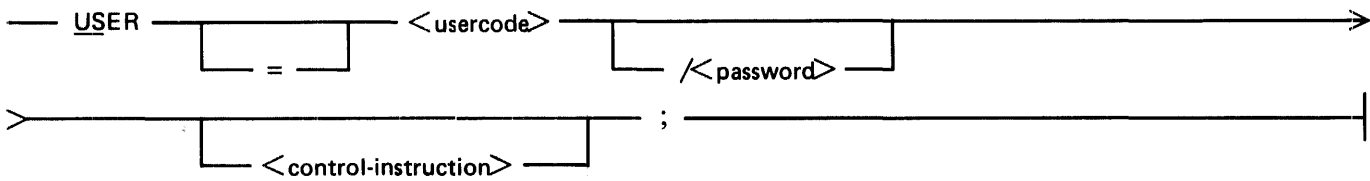
7404 UL MTA 3;

USER

The USER system command invokes the MCP file security system mechanism in the MCP and its associated naming convention.

The USER system command causes the MCP to verify <usercode> and <password> against the directory of valid usercodes and associated passwords in the (SYSTEM)/USERCODE file on the SYSTEM disk. The validated usercode is carried with <control-instruction> and is used to apply the MCP file security naming convention to any subsequent file-identifier references.

Syntax:



Semantics:

usercode

This field must contain a valid usercode contained in the (SYSTEM)/USERCODE file.

password

This field must contain a valid password that belongs to <usercode> and is contained in the (SYSTEM)/USERCODE file.

control-instruction

This field can contain any valid MCP command.

Examples:

Command	Response
USER SITE/PRIV PD =/=;	PD= MASTER/(SITE)/FILE1 PD= MASTER/(SITE)/FILE2 PD= MASTER/(SITE)/FILE3 END PD
PD (SITE)/=;	PD= MASTER/(SITE)/FILE1 PD= MASTER/(SITE)/FILE2 PD= MASTER/(SITE)/FILE3 END PD
US STUDENT/JK EX TESTPROG PR 3; PD (FINANCE)/CHECKREG;	(STUDENT)/TESTPROG =5 BOJ ... PD=(FINANCE)/CHECKREG END PD
REMOVE (FINANCE)/CHECKREG;	(FINANCE)/CHECKREG NOT REMOVED- SECURITY ERROR
US FINANCE/VP REMOVE CHECKREG;	"CHECKREG" REMOVED

WD (Display MCP Date)

The WD system command causes the MCP to display the current date maintained by the MCP.

The date is displayed by the MCP in the following format.

DATE = <month>/<day>/<year> <name-of-day>
(JULIAN DATE = <year><julian-day>)

Syntax:

— WD —————
 └──┬──┘
 ; —┘

Example:

Command	Response
WD;	DATE = 01/27/81 TUESDAY (JULIAN DATE = 81027)

WFL (Work Flow Language) Command

The WFL system command specifies that WFL syntax is used to evaluate the command that follows it. The WFL system command does not need to precede a command when the WFL option is set or when the command is only available in the WFL language.

The WFL system command resolves the ambiguity of four commands that have the same spelling in control card and WFL syntax. These commands are listed below and are marked with an asterisk. For example, if the WFL option is reset, then the WFL system command may precede the CHANGE command so that WFL syntax is used instead of control card syntax.

Syntax:

— WFL <system-command-in-WFL-syntax> —|

Semantics:

system-command-in-WFL-syntax

The valid WFL commands are:

*CHANGE	*REMOVE	SECURITY
PASSWORD	START	BEGIN JOB
*MODIFY	RUN	*COMPILE

The commands with an asterisk have different syntax diagrams when preceded by the CC System Command or when the WFL System Option is reset forcing the control card syntax to be used.

For a description of the WFL syntax of these commands and information of the WFL system command, refer to the B 1000 Systems Work Flow Language (WFL) Language Manual.

Examples:

WFL REMOVE A/B ON YOURPACK, (USER)8;

WFL CHANGE A/B TO C/B, D/L TO H FROM MYPACK;

WM (Display Current MCP and Interpreter)

The WM system command inquires which MCP and related system software are currently being used.

The following is the format of the message display by the MCP in response to the WM system command.

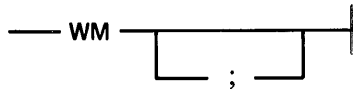
```
MCP = "<mcp-name>" VERSION <version> INTERP = "<interpreter-name>"
GISMO = "<gismo-name>" INIT = "<initializer-name>" MICRO-MCP =
"<micro.mcp-name>" NETWORK.CONTROLLER = "<network-controller-name>"
USERCODE.FILE = "<usercode-file-name>" .
```

```
<system-style-description> <devices-online>
THE FOLLOWING SOFTWARE OPTIONS ARE BEING USED:
<software-options-list> .
```

```
THE FOLLOWING MEASUREMENT TOOLS ARE BEING USED:
<software-measurement-tools-list> .
```

Disk transformation information is displayed on B 1995 systems when the disk order has been changed with the DD command.

Syntax:



Example:

Command	Response
WM;	MCP = MCP12/0724A VERSION 12.0.0 INTERP = SDL2/INTERP3M GISMO = GISMO1208/REL0716 MICRO-MCP = MMCP1206/REL0724 NETWORK CONTROLLER = COMBINED/NC_JUL3 MCS <NULL>

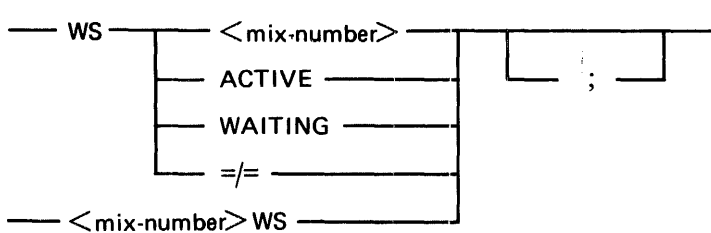
WS (Display Schedule)

The WS system command displays the status of programs in the schedule.

The response to the WS system command has the following format.

```
<mix-number> <program-name> WILL USE <integer>KB, SCHED PR =
<priority>, IN FOR <hours>:<minutes>:<seconds>.<tenths-of-a-second>,
<waiting-reason>
```

Syntax:



Semantics:

mix-number

This field, which must contain a mix number, specifies the program to interrogate.

ACTIVE

The ACTIVE keyword causes all the program names in the active schedule to be displayed.

WAITING

The WAITING keyword causes all the program names in the waiting schedule to be displayed.

=/=

The =/= key symbol causes all program names in the active and waiting schedules to be displayed.

Examples:

Command	Response
WS 40;	ALPHA = 4 NEEDS 8 KB PR = 4 IN FOR 00:008:37.4
WS ACTIVE	ACTIVE SCHEDULE EMPTY

WT (Display MCP Time)

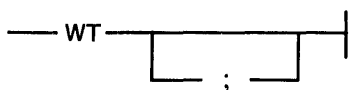
The WT system command causes the MCP to display the current time.

The following format is displayed in response to the WT system command.

TIME = <hours>:<minutes>:<seconds>.<tenths-of-a-second>

The time displayed is based on a 24-hour clock. For example, 08:10:25.3 AM is displayed as 08:10:25.3, and 08:10:25.3 PM is displayed as 20:10:25.3.

Syntax:

— WT — 

Example:

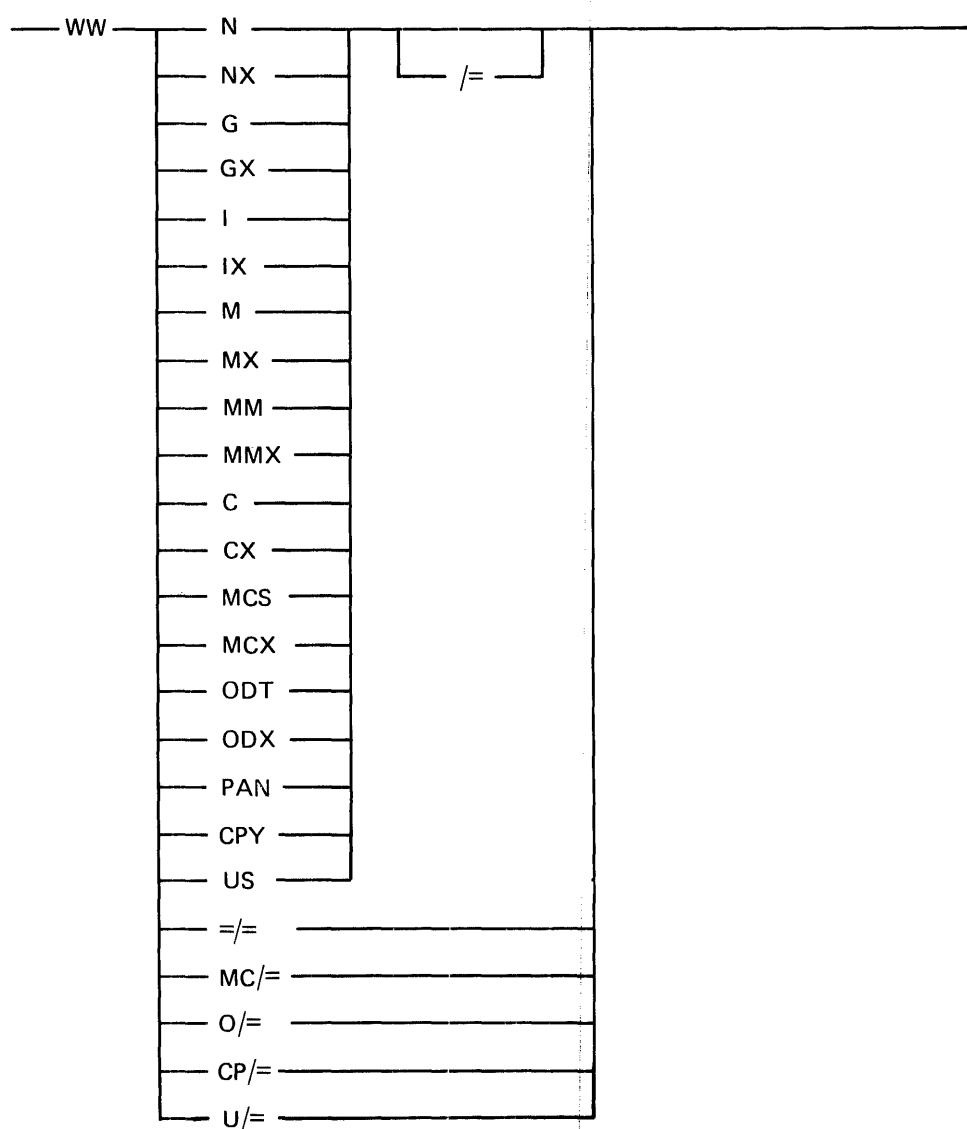
Command	Response
WT;	TIME = 09:46:39.9

WW (List Contents of the Name Table)

The WW system command lists the various types of system software/firmware in the MCP Name Table.

Refer to the clear/start program in the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the MCP Name Table entries.

Remote ODT syntax:



Semantics:

N
The N key symbol causes the current Name Table entry in the N (SYSTEM/INIT program) slot to be displayed. If the optional /= key symbol is included, the entry in the NX (experimental SYSTEM/INIT program) slot is also displayed.

NX

The NX keysymbol causes the current Name Table entry in the NX (experimental SYSTEM/INIT program) slot to be displayed. The optional /= keysymbol has no effect.

G

The G keysymbol causes the current Name Table entry in the G (GISMO firmware) slot to be displayed. If the optional /= keysymbol is included, the entry in the GX (experimental GISMO firmware) slot is also displayed.

GX

The GX keysymbol causes the current Name Table entry in the GX (experimental GISMO firmware) slot to be displayed. The optional /= keysymbol has no effect.

I

The I keysymbol causes the current Name Table entry in the I (SDL2/INTERP interpreter) slot to be displayed. If the optional /= keysymbol is included, the entry in the IX (experimental SDL2 interpreter) slot is also displayed.

IX

The IX keysymbol causes the current Name Table entry in the IX (experimental SDL2 interpreter) slot to be displayed. The optional /= keysymbol has no effect.

M

The M keysymbol causes the current Name Table entry in the M (MCPII program) slot to be displayed. If the optional /= keysymbol is included, the entry in the MX (experimental MCPII program) slot is also displayed.

MX

The MX keysymbol causes the current Name Table entry in the MX (experimental MCPII program) slot to be displayed. The optional /= keysymbol has no effect.

MM

The MM keysymbol causes the current Name Table entry in the MM (MCPII/MICRO-MCP program) slot to be displayed. If the optional /= keysymbol is included, the entry in the MMX (experimental MCPII/MICRO-MCP program) slot is also displayed.

MMX

The MMX keysymbol causes the current Name Table entry in the MMX (experimental MCPII/MICRO-MCP program) slot to be displayed. The optional /= keysymbol has no effect.

C

The C keysymbol causes the current Name Table entry in the C (network controller) slot to be displayed. If the optional /= keysymbol is included, the entry in the CX (experimental network controller) slot is also displayed.

CX

The CX keysymbol causes the current Name Table entry in the CX (experimental network controller) slot to be displayed. The optional /= keysymbol has no effect.

MCS

The MCS keysymbol causes the current Name Table entry in the MCS (MCS program) slot to be displayed. The optional /= keysymbol has no effect.

MCX

The MCX keysymbol causes the current Name Table entry in the MCX (experimental MCS program) slot to be displayed. The optional /= keysymbol has no effect.

ODT

The ODT keysymbol causes the current Name Table entry in the ODT (SYSTEM/ODT program) slot to be displayed. The optional /= keysymbol has no effect.

ODX

The ODX keysymbol causes the current Name Table entry in the ODX (SYSTEM/ODT program) slot to be displayed. The optional /= keysymbol has no effect.

PAN

The keysymbol PAN identifies the current name table entry in the PAN (SYSTEM/PANDA program) slot to be displayed. The optional /= keysymbol has no effect.

CPY

The CPY keysymbol causes the current Name Table entry in the CPY (SYSTEM/COPY program) slot to be displayed. The optional /= keysymbol has no effect.

US

The US keysymbol causes the current Name Table entry in the US ((SYSTEM)/USERCODE file) slot to be displayed. The optional /= keysymbol has no effect.

=/=

The =/= keysymbol causes the current Name Table entry in all the Name Table slots to be displayed.

MC/=

The MC/= keysymbol causes the current Name Table entries in the MCS (MCS program) slot and the MCX (experimental MCS program) slot to be displayed.

O/=

The O/= keysymbol causes the current Name Table entries in the ODT (SYSTEM/ODT program) slot and the ODX (experimental SYSTEM/ODT program) slot to be displayed.

CP/=

The CP/= keysymbol causes the current Name Table entry in the CPY (SYSTEM/COPY program) slot to be displayed.

U/=

The U/= keysymbol causes the current Name Table entry in the US ((SYSTEM)/USERCODE file) slot to be displayed.

Examples:

Command	Response
WW M/=;	M = MCP11 MX = MCP11/OLD
WW US;	US = (SYSTEM)/USERCODE
WW CX;	CX = SYSTEM/CONTROLLER

WY (Mix and Status Interrogation)

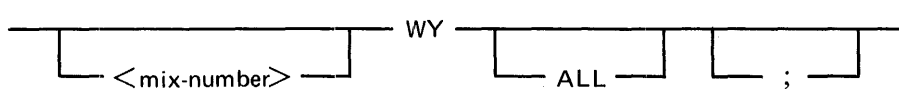
The WY system command displays the priority and current status of one or all programs in the mix. The WY system command functions identically to the MX system command.

If the WY system command is prefaced with a mix number, only information about that specific program is displayed. If the mix number is omitted, information about all visible tasks in the mix is displayed. Refer to the IV system command for information concerning task invisibility. If the optional ALL keyword is included, information about all tasks, including invisible tasks, is displayed.

If a program is awaiting operator action, the information displayed includes the action alternatives available to the operator.

If the MPRI MCP option is set, the processor and memory priorities are displayed instead of the priority number.

Syntax:



Semantics:

mix-number

This field must contain a valid mix number of a program currently in the mix and specifies the task in which to interrogate the priority and status.

ALL

The ALL keyword specifies that information about all tasks, including invisible tasks, is to be displayed.

The following examples assume that the MPRI MCP option is set. If the MPRI MCP option is not set, the priority number is displayed instead of the processor and memory priorities.

Examples:

Command	Response
WY;	SMCS = 308 PP = 13, MP = 13 "WAIT" STATUS CANDE = 321 PP = 12, MP = 12 "WAIT" STATUS RD = 310 PP = 11, MP = 11 "WAIT" STATUS SYCOM : PS = 311 PP = 9, MP = 9 "WAIT" STATUS 6 PROGRAMS RUNNING...END MX/WY.
WY ALL;	SYSTEM/ODT = 304 PP = 15, MP = 15 EXECUTING SYSTEM/CONTROLLER = 303 PP = 15, MP = 15 EXECUTING SMCS = 308 PP = 13, MP = 13 "WAIT" STATUS CANDE = 321 PP = 12, MP = 12 "WAIT" STATUS RD = 310 PP = 11, MP = 11 "WAIT" STATUS SYCOM : PS = 311 PP = 9, MP = 9 "WAIT" STATUS 6 PROGRAMS RUNNING...END MX/WY.
321WY;	CANDE = 321 PP = 12, MP = 12 "WAIT" STATUS

XC (Remove Segments Temporarily)

The XC system command temporarily removes contiguous disk segments from the MCP tables of available disk space.

The disk space to be removed must be available in order to be removed. If any portion of the space is occupied, for example with a file or disk file header, the MCP rejects the request with the following message.

REQUESTED SEGMENT NOT REMOVED – NOT AVAILABLE

If the disk is a SYSTEM disk, the disk space is returned after the next clear/start operation. If the disk is a user disk, the disk space is returned after the user disk has been powered down with the PO system command.

Remote ODT syntax:

```

XC <unit-mnemonic> <integer-2> <integer-3>
  |
  | DKx <integer-1>
  |
  ;
  
```

Semantics:

integer-1

This field, which must contain an integer, specifies the electronics unit (EU) for the head-per-track disk.

integer-2

This field, which must contain an integer, specifies the beginning segment address, and can be expressed in any format. If the operation is being performed on a disk cartridge (DCx), and the beginning segment address is not the address of the first segment in a track, the MCP automatically adjusts it backward to the beginning of the track.

integer-3

This field, which must contain an integer, specifies the number of disk segments to be removed. If the operation is being performed on a disk cartridge, the number of segments removed is a multiple of the entire track. The MCP makes the necessary adjustments, both in starting address and number of segments.

Examples:

Command	Response
XC DKA 0 200 1000;	DISK SPACE REMOVED FROM @EE0000C8@ THRU @EE00004AF@
XC DPC @46@ 30;	DISK SPACE REMOVED FROM @F22000046@ THRU @F22000063@

XD (Remove Segments Permanently)

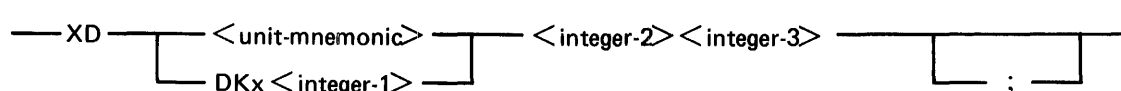
The XD system command permanently removes contiguous disk segments from the MCP tables of available disk space.

The disk space to be removed must be available in order to be removed. If any portion of the space is occupied, for example with a file or disk file header, the MCP rejects the request with the following message.

REQUESTED SEGMENT NOT REMOVED – NOT AVAILABLE

To return the removed disk segments, a disk initialization (for user disks) or coldstart operation (for head-per-track disks) must be performed.

Remote ODT syntax:



Semantics:

integer-1

This field, which must contain an integer, specifies the electronics unit (EU) for the head-per-track disk.

integer-2

This field, which must contain an integer, specifies the beginning segment address, and can be expressed in any format. If the operation is being performed on a disk cartridge (DCx) and the beginning segment address is not the address of the first segment in a track, the MCP automatically adjusts the address backward to the beginning of the track.

integer-3

This field, which must contain an integer, specifies the number of disk segments to be removed. If the operation is being performed on a disk cartridge, the number of segments removed is a multiple of the entire track. The MCP makes the necessary adjustments, both in starting address and number of segments.

Examples:

Command	Response
XD DKA 0 200 1000;	DISK SPACE REMOVED FROM @EE0000C8@ THRU @EE00004AF@
XD DPC @46@ 30;	DISK SPACE REMOVED FROM @F22000046@ THRU @F22000063@

SECTION 6

NETWORK CONTROLLER OPERATIONS

NETWORK CONTROLLER EXECUTION

After the network controller program is compiled, it can be executed in two ways: explicitly or automatically.

The network controller can be explicitly executed using the following MCP control instruction:

```
EXECUTE <network-controller-program-identifier>;
```

To execute the network controller automatically by the MCP, the name of the network controller must be specified to the MCP. This is done by placing the identifier of the network controller program into the C entry, or slot, of the MCP Name Table using the following MCP command:

```
CM C <network-controller-program-identifier>
```

An experimental network controller can be placed in the CX entry, or slot, of the MCP Name Table using the following MCP command:

```
CM CX <network-controller-program-identifier>;
```

NETWORK CONTROLLER PRIORITY

For optimum response time for the remote programs, it is strongly suggested that the network controller run at a higher priority than other tasks in the mix. The network controller is automatically assigned a priority of 15 by the NDL compiler and the invisible bit is set.

NETWORK CONTROLLER PROGRAM SWITCHES

The network controller program switches are described in the following paragraphs.

Program Switch 2

Program switch 2 controls the network controller handling of a data communication line that is hung or not responding properly. In all cases of line hangs, a message is displayed on the ODT. The following action is then taken depending on the value of program switch 2.

- 0 A program dump is taken and the line is restarted.
- 1 The line is restarted.
- 2 The line is left in a hung status.
- 3-15 The network controller is suspended until the operator responds by entering the NC ODT command.

Program Switch 3

Program switch 3 controls the initial accumulation of statistics by the network controller. The following action is taken depending on the value of program switch 3.

0	The statistics function is controlled by the STATISTICS declaration in the network controller symbolic code.
1	Unconditionally initiate accumulation of statistics from beginning of job.
2	Do not initiate accumulation of statistics regardless of the STATISTICS declaration in the network controller symbolic code.
3-15	Same as for program switch 3 equal to 0.

Program Switch 7

If the network controller program switch 7 is set to 1, the IOLOG debugging facility is initiated at beginning of job.

NETWORK CONTROLLER INPUT COMMANDS

Input commands direct the network controller to perform specific functions. The system commands are described in the following paragraphs. All ODT system commands are prefixed by the MCP NC command, or optionally, by the network controller mix number and the MCP AX or AC keywords.

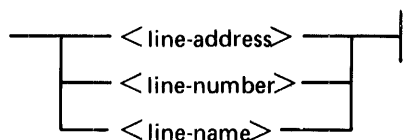
Common Syntax

The following syntactic items described in this section are used in the specification of the syntax of one or more network controller commands. The items are presented alphabetically for quick reference.

<line-id>

The <line-id> is used to specify the line for which the command is to be performed.

Syntax:



Semantics:

<line-address>

This field can contain the port, channel, and adapter declared for the line in the network controller symbolic code. The format of the <line-address> is <port>:<channel>:<adapter>.

<line-number>

This field can contain the logical number of the line as declared in the network controller symbolic code.

<line-name>

This field can contain the symbolic name assigned to the line in the network controller symbolic file.

Examples:

2:0:0

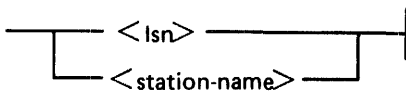
1

TDLINE4

<station-id>

This field can contain the station for which the command is to be performed.

Syntax:



Semantics:

lsn

The <lsn> field specifies logical station number of the station.

station-name

The <station-name> field specifies symbolic name assigned to the station in the network controller symbolic file.

Examples:

13

AARDVARK

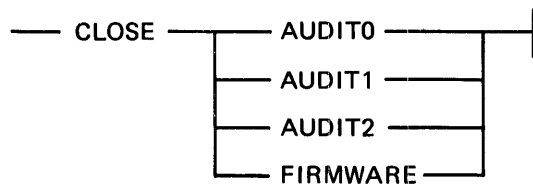
ODT COMMANDS

The following paragraphs describe the NC ODT commands.

CLOSE Command

The CLOSE network controller command closes the specified network controller audit file or the currently open firmware file.

Syntax:



Semantics:

AUDIT0, AUDIT1, and AUDIT2

The AUDIT0, AUDIT1, and AUDIT2 keywords specify which audit file to close. AUDIT0 specifies the first audit file declared in the network controller symbolic file. AUDIT1 specifies the second audit file declared and AUDIT3 specifies the third audit file declared.

FIRMWARE

The FIRMWARE keyword causes the currently open firmware file to be closed.

If the firmware file is open in the network controller code file, the sole effect of the CLOSE FIRMWARE command is a temporary saving of memory. If the firmware file is open external to the network controller code file, the CLOSE FIRMWARE command releases the external firmware file for removal or replacement.

Examples:

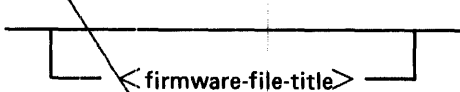
CLOSE AUDIT0

CLOSE FIRMWARE

CREATE Command

The CREATE network controller command creates a new firmware file for the B 1990 from the firmware stored in the currently running network controller.

Syntax:

— CREATE FIRMWARE 

Semantics:

<firmware-file-title>

This field specifies the file title of the new firmware file that is to be created.

If the <firmware-file-title> is omitted, the default file title of the created firmware file is SYSTEM/MLFIRMWARE on the system disk.

Examples:

CREATE FIRMWARE

CREATE FIRMWARE NEW/FIRMWARE

CREATE FIRMWARE SYSTEM/MLFIRMWARE ON DATACOMM

DATARATE Command

The DATARATE network controller command sets or resets the CCITT data rate for the B 1990 system. Since the B 1990 system obtains the data rate value from the data communication controls and line adapters, the DATARATE command needs only the name of the line adapter or the hardware address as specified in the network controller program.

Syntax:

```
— RESET ——— DATARATE <line-id> ———|
— SET   ———|
```

Semantics:

RESET

The RESET keyword causes the CCITT data rate to be reset.

SET

The SET keyword causes the CCITT data rate to be set.

Examples:

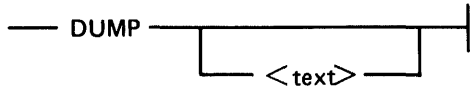
```
SET DATARATE TDLINE1
```

```
RESET DATARATE 2:0:0
```

DUMP Command

The DUMP network controller command causes a program dump of the network controller to be created.

Syntax:



Semantics:

text
This field specifies the message that is to appear in the analyzed dump and can be used to identify the reason for the dump.

Examples:

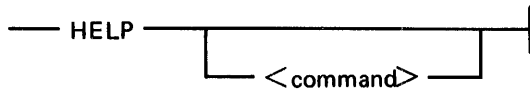
DUMP

DUMP Beware of rainy days and Mondays.

HELP Command

The HELP network controller command displays a list of available network controller commands or a syntax diagram of the specified command.

Syntax:



Semantics:

command

The field specifies the command for which the syntax diagram is to be displayed.

Examples:

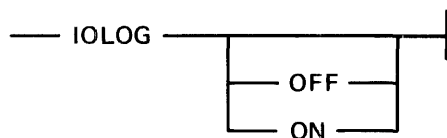
HELP

HELP STATUS

IOLOG Command

The IOLOG network controller command initiates or queries the setting of the IOLOG facility.

Syntax:



Semantics:

OFF

Specifying the OFF keyword causes the IOLOG facility to be turned off.

ON

Specifying the ON keyword causes the IOLOG facility to be initiated.

Pragmatics:

The IOLOG facility monitors the data communication line activity, writing the information to a disk file labeled DC/AUDITFILE.

If no keyword is specified in the IOLOG command, the current status of the IOLOG facility is returned.

Examples:

IOLOG ON

IOLOG OFF

IOLOG

MAKE Command

The MAKE network controller command makes a station that is not controlled by a Message Control System (MCS) ready or not ready.

Syntax:

— MAKE STATION <station-id> ———— NOT ———— READY ————

Semantics:

NOT

The NOT keyword specifies that the station is to be made not ready.

Examples:

MAKE STATION AARDVARK NOT READY

MAKE STATION 29 NOT READY

MAKE STATION 72 READY

MAKE STATION WOMBAT READY

QUIT Command

The QUIT network controller command terminates the network controller.

If the QUIT command is entered, the network controller terminates after all files are closed, regardless of the state of the line and the number of messages queued for delivery to stations.

Syntax:

— QUIT —|

Example:

QUIT

IF operating from CX slot on
a B1990 Re NC in said
CX slot will be re-executed

RELOAD LINE Command

The RELOAD LINE network controller command reloads firmware loaded into the requested line for a MLC-4 quad line adapter.

When the RELOAD LINE command is entered, all four lines on the quad adapter are interrupted if they are under the control of the network controller. After the firmware is reloaded, the relevant line configurations are loaded into the line adapter and a RESTART LINE command is automatically performed for each line.

The RELOAD LINE command is not accepted for multiline controls other than the MLC-4 quad line adapter.

Syntax:

— RELOAD LINE <line-id> —|

Examples:

RELOAD LINE TDLINE

RELOAD LINE 2:0:1

RELOAD LINE 5

RELEASE LINE Command

The RELEASE LINE network controller command releases a line from the control of the network controller for use with another data communication program, such as HASP or RJE programs.

Syntax:

— RELEASE LINE <line-id> —|

Pragmatics:

When a RELEASE LINE command is entered, the adapter is unconditionally cleared, marked as not present, and the line released to the system.

Examples:

RELEASE LINE TDLINE5

RELEASE LINE 2:0:4

RELEASE LINE 5

RESTART LINE Command

The RESTART LINE network controller command clears line adapters that are suspected of an improper (or lack of) response and changes the line address of the line.

Syntax:

```
— RESTART LINE <line-id> _____ |
                                     |
                                     |_____|
                                     <new-address>
```

Semantics:

new-address

This field specifies the port, channel, and adapter of the new line address. <new-address> has the following format:

```
<port> : <channel> : <adapter>
```

Pragmatics:

When a RESTART LINE command is entered, the network controller unconditionally clears the line adapter. If the line was marked as not present, meaning that it was never accessed or that it was released using the RELEASE LINE command, the network controller tests for the presence and the line adapter type of the adapter.

The network controller then takes the appropriate action depending on what the status of the line was prior to the entry of the RESTART LINE command.

1. If the line was originally marked as not present, disconnected ready, waiting for dialout, or waiting for a LOGICALACK, no further action is taken.
2. If the line was originally in ring I/O status (waiting for an incoming call), the line is returned to ring I/O status to wait for an incoming call.
3. If the line was in the process of dialing out, the dialout operation is resumed.
4. If the line was in any other state, the line control procedure is entered with LINE(CONTROL KEY) equal to 1.

<new-address> option allows the address of the line to be changed. <new-address> option is only valid if the line is in a released/not present status.

Examples:

```
RESTART LINE TDLINE5
```

```
RESTART LINE 2:0:4
```

```
RESTART LINE 5
```

```
RESTART LINE LINE13 2:0:7
```

STANDBY Command

The STANDBY network controller command sets or resets the CCITT standby feature for the B 1990.

Syntax:

```
— RESET ——— STANDBY <line-id> ———|  
— SET ———|
```

Semantics:

RESET

The RESET keyword causes the CCITT standby feature to be reset.

SET

The SET keyword causes the CCITT standby feature to be set.

Examples:

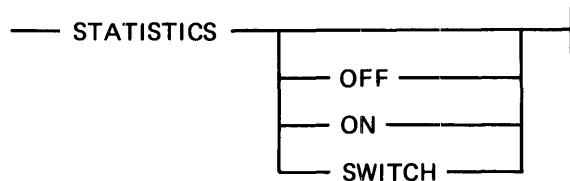
```
RESET STANDBY 2:0:10
```

```
SET STANDBY LINE4
```

STATISTICS Command

The STATISTICS network controller command displays the current status of the statistics function, initiates or terminates the statistics function, and closes the current statistics file.

Syntax:



Semantics:

OFF

The OFF keyword causes the statistics function to be terminated and the statistics file to be closed.

ON

The ON keyword causes the statistics function to be initiated.

SWITCH

The SWITCH keyword causes the current statistics file to be closed and a new statistics file to be opened.

Examples:

STATISTICS

STATISTICS OFF

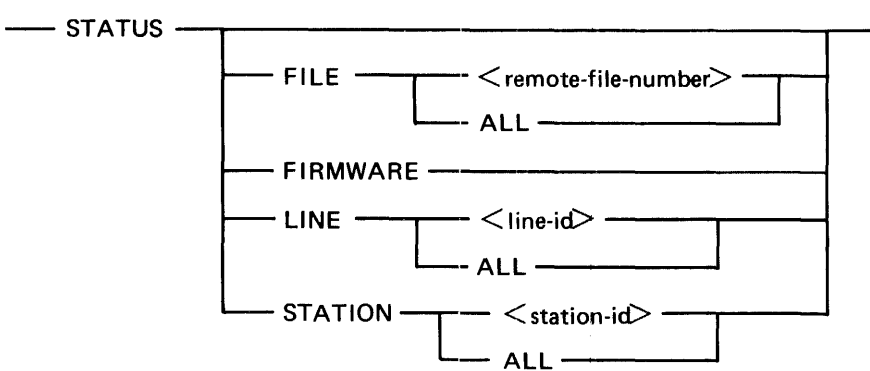
STATISTICS ON

STATISTICS SWITCH

STATUS Command

The STATUS network controller command displays the version of the network controller, line status, station status, file status and firmware status.

Syntax:



Semantics:

remote-file-number

This specifies the logical number of the file as it is opened by the network controller.

FILE ALL

The FILE ALL option causes the status of all open remote files to be displayed.

LINE ALL

The LINE ALL option causes the status of all the lines declared in the network controller to be displayed.

STATION ALL

The STATION ALL option causes the status of all the stations declared in the network controller to be displayed.

Pragmatics:

When the status of a line is requested, the network controller displays the line status, the adapter type, and the number of messages queued for that line.

When the status of a station is requested, the network controller displays the station type, the number of messages queued, and the primary and secondary file status for the station.

When the status of a file is requested, the network controller displays information about the remote file number requested. If the ALL option is specified, the status of all open remote files is displayed.

When the status of the firmware is requested, the network controller displays the relevant creation dates for the firmware in the network controller code file.

If no options are specified, the release version of the SYSTEM/CONTROLLER program is displayed.

Examples:

STATUS

STATUS FILE 4

STATUS FILE ALL

STATUS FIRMWARE

STATUS LINE 5

STATUS LINE TDLINE5

STATUS LINE 2:0:4

STATUS LINE ALL

STATUS STATION 22

STATUS STATION PLATYPUS

STATUS STATION ALL

DATA COMMUNICATIONS ODT

The SYSTEM/ODT program can operate with either the multiline control or the ODT control. This feature of an ODT running through the multiline control is referred to as a data communication ODT.

The data communication ODT implementation is mandatory for the B 1990 systems and optional for all other B 1000 systems.

To run a data communication ODT, it is required that an MT 985 terminal be configured to run two pages of 1920 characters each. The MT 985 terminals have Mark 3.8 (or later) firmware installed and firmware location A0 must be set.

NOTE

Due to changes in the Mark 3.8 firmware in the MT 985 terminal, an explicit exit forms mode sequence (4"27" X) must be sent for every message intended to clear a screen that is already in forms mode. Prior to the Mark 3.8 firmware, or if firmware location A0, bit 1 is reset, the receipt of a 'SOH' character that precedes a data communication message takes the station out of forms mode.

For user programs that utilize forms mode to function correctly on a data communication ODT, the exit forms mode sequence should precede each message.

DIFFERENCES BETWEEN DATA COMMUNICATION AND ODT-CONTROL ODT Device

The data communication ODT behaves functionally the same as an ODT device connected to an ODT control with the following exceptions:

1. All ODT output is directed to page 2 of the ODT. Page 1 is used for register information on the B 1990 systems when the system is interrupted or halted.
2. The SPCFY key is used to change pages on the data communication ODT. The CTRL → (right arrow) key sequence should never be used to change pages on a data communication ODT; unpredictable results can occur as a result of using the CTRL → (right arrow) key sequence to change pages. At network controller beginning of job, the SPCFY key must be depressed three times to initially change pages; after the initial change of pages, the SPCFY key need only be depressed once.

Using the Data Communications ODT as a Terminal

As a consequence of the implementation of the data communication ODT, it is now possible to run user programs on the ODT with the following restrictions:

1. Only page 1 of the terminal is available for use.
2. The SPCFY key cannot be used by the program.
3. If forms mode is to be used, the Mark 3.8 (or later) firmware in the MT 983 terminal must be used. In addition, firmware location A0/1 must be set.

The following procedures are used to run the CANDE program on the data communication ODT. For the purposes of this discussion, it is assumed that the SMCS program is the Supervisory Message Control System (MCS) and that the SMCS program is currently running in the mix.

1. Attach the data communication ODT to the SMCS program using the SMCS ATTACH command as follows:

```
<SMCS mix number> AX ATTACH SYSTEMODT
```

2. Change to page 1 of the data communication ODT by depressing the SPCFY key on the ODT keyboard. All user programs are run on page 1. When the data communication ODT is attached to a user program such as the CANDE program, all input from page 1 of the data communication ODT is directed to the user program and all input from page 2 is directed to the SYSTEM/ODT program. If the CTRL → (right arrow) or CTRL ← (left arrow) control sequences are used to change pages, the network controller may direct the input to the wrong program.
3. Log on to the SMCS program by entering the following.

```
USER APTERYX/KIWI
```

4. Sign on to the CANDE program by entering the following.

```
SIGN ON CANDE
```

The user can now edit a work file and perform other CANDE functions.

The user can also execute programs that open a remote file that is file-equated to the system ODT as follows. The SYSTEM/CONFIGURE program is used as an example:

1. Execute the SYSTEM/CONFIGURE program as follows:

```
EXECUTE SYSTEM/CONFIGURE FILE REMOTE STATIONS SYSTEMODT;
```

2. Change to page 1 of the data communication ODT by depressing the SPCFY key on the ODT keyboard. All user programs are run on page 1.
3. Enter the appropriate specifications to the SYSTEM/CONFIGURE program.

The above examples can be applied to any user program. If it is desired to enter ODT commands, the user can switch to page 2 (using the SPCFY key) and enter the ODT command. All ODT output is directed to page 2.

SECTION 7

SYSTEM OUTPUT MESSAGES

The B 1000 system communicates information to the operator through the Operator Display Terminal (ODT). This information is communicated in the form of system output messages. System output messages are originated by the MCP in response to an operator inquiry or action (for example, in response to the BB, MX, or TI system commands), or in response to a specific requirement of an executing program (for example, the FILE NOT PRESENT or NO MEMORY messages). Output messages can also be originated by executing programs using DISPLAY or ACCEPT messages. These messages are preceded by a percent sign (%) character, in order to distinguish them from system messages concerning a requirement of the program.

All system output messages are indented one space to enable the operator to distinguish them from system commands. Continuation lines of system output messages are indented two spaces.

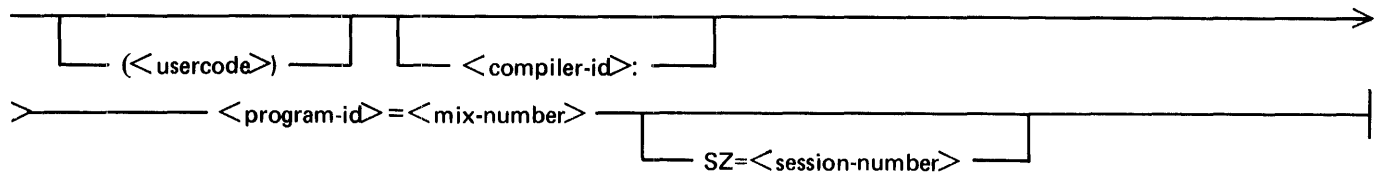
OUTPUT MESSAGE SYNTAX

The following paragraphs describe the syntax used in this section to describe several common portions of system output messages.

<task-specifier>

The <task-specifier> portion of a system output message is used by the MCP to identify the program to which the message applies. Messages which reference programs are always preceded by the <task-specifier>. Messages of a general system nature which do not concern a specific program have no <task-specifier> prefix.

The format of the <task-specifier> is as follows:



The <usercode> portion of the <task-specifier> is included if the program was executed with a usercode/password (refer to File Security in section 1 for details).

The <compiler-id> portion of the <task-specifier> is included if the COMPILE program control instruction is used. In this case, the <program-id> portion is the identification of the program being compiled, and the <compiler-id> portion specifies the compiler being used.

The <mix-number> portion of the <task-specifier> is present in all system output messages referencing an executing program, and is the means used to relate system commands to the programs they reference. The <mix-number> is assigned to a program when it is scheduled for execution by the MCP, and remains the same all during the life of the job (scheduling, beginning of job, execution, and end of job). The <mix-number> is the only unique entity that identifies a program while it is scheduled or executing, since multiple copies of the same program (by name) can be in the schedule or the mix concurrently. The <mix-number> assigned by the MCP is incremented by 1 starting with the number 1, which is the first task executed after a coldstart operation, and increases to 9999, at which time a task is assigned the number 1 again.

The <session-number> portion of the <task-specifier> is included only if the referenced program is spawned by another program, and the control string included an SZ command specifying a <session-number>.

NOTE

There are some situations (usually when the system is thrashing due to memory saturation) in which the <task-specifier> consists only of the <mix-number>, due to lack of available memory for the MCP to obtain the <compiler-id> or <program-id>. For example, the message:

3255 NO MEMORY

can be used in place of the following message in a thrashing situation:

COBOL : USER/TEST/COMPILE = 3255 NO MEMORY

<priority>

The <priority> portion of a system output message assumes one of the following forms:

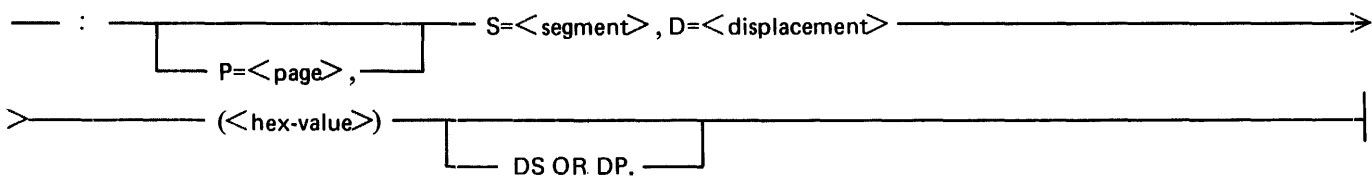
- PRIORITY=<processor-priority> —|
- PP=<processor-priority>, MP=<memory-priority> —|

The second form is used only when Level Three (Priority Memory Management) of the MCP Memory Management System is in effect (that is, the MPRI MCP option is set).

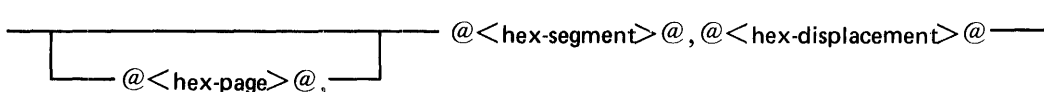
<abort-reference>

The <abort-reference> portion of a system output message is included when the DISP MCP option is set. It describes an error condition in a program that caused an abnormal termination.

The format of the <abort-reference> is:



<page>, <segment>, and <displacement> are decimal integers specifying the code segment address of the NEXT INSTRUCTION POINTER, which is the instruction immediately following the one in which the error condition was detected. <hex-value> specifies the hexadecimal equivalent of the NEXT INSTRUCTION POINTER, as follows:

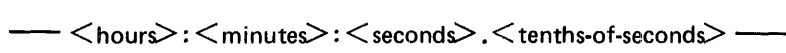


The DS OR DP portion of the <abort-reference> is included only if the TERM and TRMD MCP options are reset. In this case, an explicit DS or DP system command must be entered by the operator in order to complete termination of the program. If either the TERM or TRMD MCP option is set, the DS OR DP portion is not included, and termination of the program is automatically performed by the MCP (including a memory dump of the program, if the TRMD MCP option is set).

The <page> and <hex-page> portions of the <abort-reference> are included only if the abnormally terminated program was written in NDL, SDL, SDL2, UPL, or UPL2 (the only languages having a paged code segment dictionary).

<time>

The current time-of-day is displayed in certain system output messages, and is always in the following format:



The time-of-day is always maintained in military format, based on a 24-hour clock. For example, 08:10:25.3 is 08:10:25.3 AM and 20:10:25.3 is 08:10:25.3 PM.

<date>

The current date is displayed in certain system output messages, and is always in the following format:

<month>/<day>/<year> <name-of-day> (JULIAN DATE = <year> <julian-day>)

SECTION 8 SYSTEM HALTS

When a software-controlled halt occurs on a B 1000 system, various registers are used to display information about the halt. The most important of these is the L register, which is considered the primary halt definition. Some halts display further descriptive information in other registers (usually X, Y and T).

The L register is functionally divided into two portions.

1. The left-most 16 bits (bits 0-15) describe the specific program or routine that halted, as follows:

@0000@	SDL2 and SDL Interpreter STATE light ON – MCP STATE light OFF – Normal-state SDL program
@0200@	Micro MCP
@0D00@	GISMO
@00F0@	CLEAR/START
@000F@	SYSTEM/INIT

2. The right-most 8 bits (bits 16-23) give the halt identification. This portion of the halt code is dependent upon the specific routine that halted (given in bits 0-15).

SDL2 AND SDL INTERPRETER HALTS (L = @0000xx@)

@01@	Evaluation/Program Pointer Stack Overflow.
@02@	Control Stack Overflow.
@03@	Name/Value Stack Overflow.
@04@	Remapped area has insufficient length.
@05@	Invalid Parameter passed to a PROCEDURE.
@06@	Invalid Substring (SUBBIT or SUBSTR).
@07@	Invalid Subscript.
@08@	Invalid Value returned from a TYPED PROCEDURE.
@09@	Invalid CASE.
@0A@	Divide by Zero.
@0B@	Invalid Index. <i>uninitialized Reference Jan 66</i>
@0C@	Read out of bounds or a memory parity error.
@0D@	Invalid operator.
@0E@	Evaluation/Data Stack overflow.
@0F@	CONVERT error.
@10@	Console Halt (INTERRUPT switch).
@11@	HALT Operator. The T Register contains a further definition of the halt. Complete information on MCP halts is given below.
@12@	Write out-of-bounds.
@13@	Exponent overflow.
@14@	Exponent underflow.
@15@	Expression out of range.
@16@	Superfluous exit.
@17@	Out of memory space.
@18@	Invalid link.

@19@	Program Pointer Stack overflow.
@1A@	Integer overflow.
@1B@	Message transfer data not present.
@1C@	Message transfer invalid data template.
@1E@	Invalid Parameter in DYNAMIC Declaration.
@1F@	Invalid TRANSLATE.
@20@	Invalid subprogram type.
@21@	Reference assignment length mismatch.
@22@	Invalid operand.
@23@	Multiply result is too large.
@24@	Evaluation Stack underflow.
@25@	Invalid SET member.
@26@	Invalid Communicate.
@27@	Program Pointer Stack underflow.
@28@	Reference to an invalid variant of a record.
@29@	Attempt to de-reference a null pointer.
@2A@	Reference to a disposed dynamic variable.
@2B@	Value out of range.
@2C@	Set out of range.
@2D@	File declared in a recursive routine.
@2E@	Illegal assignment to a FOR-LOOP index.
@2F@	Attempt to change a variant with the variant in use.
@30@	Attribute error.
@31@	Uninitialized data item.
@35@	B 1870/B 1860 Console Cassette Data Error
@37@	B 1870/B 1860 RWOAM detected.

clS

SI

FIRMWARE HALTS (L = @00FOxx@ OR @000Fxx@)

@02@	A clear/start disk channel was specified in the X register, but the device on the specified channel is not a disk device (TE/TF = Device ID).
@03@	Control is not idle; that is, the control is not in status count 1 (T = Status Count).
@04@	Timeout while waiting for SERVICE REQUEST. Push START to retry.
@05@	Bad Reference Address (X = Good Reference Address, Y = Bad Reference Address, TE = Port, TF = Channel). Push START to use the good Reference Address and continue.
@06@	A control is in an erroneous Status Count following a SERVICE.REQUEST (TF = Channel).
@07@	A bad RESULT.DESRIPTOR was received from a control: the OP.COMPLETE bit (bit 0) was reset (T = RESULT.DESRIPTOR).
@08@	A SEEK TIMEOUT occurred. Push START to continue.
@0A@	A memory parity error was detected in I/O data. Push START to continue.
@0B@	A TIMEOUT occurred while waiting for OP.COMPLETE on a port device. Push START to retry.
@0C@	An I/O exception condition remained after 15 retries (T = RESULT.DESRIPTOR). Push START to retry.

	@0F@	No disk control found on the system.
	@15@	A NAME TABLE entry is empty.
	@17@	SYSTEM/DUMPFIL E disk address is zero, and a dump was requested (DUMP option is probably reset).
F0	@1E@	Checksum error. T = SOFTWARE.ID.NO (See Note below).
	@22@	Cassette data error.
	@26@	The NAME TABLE is in an incorrect format for this release of clear/start.
	@30@	TEXT input (B 1990 systems) is in error. Push START or enter correct text and push START.
	@35@	Other than system disk on-line at clear/start
	@83@	A discardable segment of GISMO required by the system hardware or specific MCP options that are set is not present in the selected GISMO. Push START to ignore the non-present segment and continue.
	@86@	Software compatibility problem. TE and TF=SOFTWARE.ID.NOs (See Note 1). If TD=1 then X=Level, Y=Level If TD=7 then X="NEEDS", Y="HAS".
	@8B@	Pseudo MAXS too large (X=pseudo MAXS, Y=maximum value on the system). Push START to use value in Y.

NOTE

Software ID Numbers:

- 1 = System Initializer
- 2 = GISMO
- 3 = Micro MCP
- 4 = MCP
- 5 = SDL2 Interpreter
- 6 = clear/start

GISMO HALTS (L=@0D00xx@)

@10@	Console Halt (INTERRUPT switch).
@12@	MCP write out of bounds. X=LIMIT.REGISTER.
@20@	(RFAC option set) Bad Reference Address returned from a control (X=Good Reference Address, Y=Bad Reference Address, TA=Channel, TE/TF=Status Count). Push START to use good Reference Address.
@21@	(RFAC option set) Irrecoverable Reference Address error (same register settings as @0D0020@ halt).
@22@	(RFAC option set) Software attempted to send a Reference Address of zero to a control.
@24@	Unknown device id returned by a control (X=Reference Address, TB=Channel).
@27@	(Debug GISMO only) A pause op was sent to an uninitialized channel (X=Reference Address, TB=Channel).
@28@	(Debug GISMO only) PAUSE or SEEK.COMPLETE problem (X=Reference Address, TB=Channel).

- @30@ INTERRUPT.QUEUE Overflow (X=number of messages attempted, Y=maximum entries permitted).
- @31@ The dispatch (INCN) register is locked, indicating that a port device is hung. (TE=Port, TF=Channel, X=Reference Address of message attempting to be sent.)
- @36@ Bad COMMUNICATE.WITH.GISMO verb (T=FA value when reading communicate).
- @37@ Bad COMMUNICATE.WITH.GISMO adverb (X=Communicate function, T=FA value when reading Communicate).
- @38@ COMMUNICATE.WITH.GISMO/GISMO.COMMUNICATE: Parameter list length error (T=FA address).
- @39@ Bad GISMO.COMMUNICATE verb (X=verb).
- @42@ A program attempted to use a deleted GISMO function (T=LIMIT.REGISTER, X=Function). Reloading the program's interpreter is warranted before pursuing the problem further.
- @44@ The MCP dispatched an I/O operation with a result descriptor that was already in process.
- @47@ The port device returned an illegal high-priority interrupt on a DISPATCH.READ micro-instruction (INCN register bit 1 set) (TE=Port, TF=Channel, X=Reference Address + 24).
- @53@ Cassette Data error at entry to GISMO (Y=STATE.FLAGS, X=LIMIT.REGISTER).
- @54@ CPU multiple bit parity error (T=ELOG register contents, Y=STATE.FLAGS, X=LIMIT.REGISTER).
- @55@ Attempted to read or write outside the physical bounds of main memory. X=LIMIT.REGISTER, Y=STATE.FLAGS. If Y equal 0, read or write occurred while GISMO was running. If Y is not equal to 0, read or write occurred while the program with the LIMIT.REGISTER in X was running.
- @60@ Service request received from a missing or ignored channel (TB = Channel).
- @A0@ (Dual Processors only and TOUT option set) CPU timeout: The processor that halted detected that the other processor timed out due to either being halted or being in a micro-loop.
- @A1@ (Dual Processors only) Invalid message sent from the slave processor.
- @A2@ (Dual Processors only) Invalid message sent from the master processor.
- @A3@ (Dual Processors only) Invalid GISMO request made by an interpreter running on the slave processor.
- @A4@ (Dual Processors only) More unblock requests that block requests were sent to the slave processor.
- @A7@ (Dual Processors only) A reset service request operation fails on the slave processor.
- @A8@ (Dual Processors only and TOUT option set) Halt when a processor has detected that the other processor has halted with the @0D00A0@ halt.
- @AA@ CONSOLE.INTERRUPT is true while waiting for response from a DCPU.DISPATCH.
- @103@ Attempted to decrement RS.BLOCK.COUNT which was already zero.

MICRO MCP HALTS (L = @0200xx@)

- @01@ There is a message in the Micro MCP INTERRUPT.QUEUE which indicates that a completed I/O is to be handled by Micro MCP, but the RESULT.DESCRIPTOR indicates that the I/O is not yet complete.
- @02@ A message in the Micro MCP INTERRUPT.QUEUE for an I/O has an erroneous value in the IO.MCP.IO field.
- @04@ A bad value for IO.MCP.IO was detected while examining an I/O descriptor.
- @05@ While examining an I/O descriptor, the Micro MCP found that the hard or soft I/O complete bits (bits 0 or 1 of the IO.M.EVENTS field) were set, yet the IO.COMPLETE bit (bit 0 of the RESULT.DESCRIPTOR) was not set.
- @06@ While examining an I/O descriptor, the Micro MCP found that the IO.COMPLETE bit (bit 0 of the RESULT.DESCRIPTOR) was set, yet the hard I/O complete bit (bit 0 of the IO.M.EVENTS) was not set.
- For halts @020001@ through @020006@: T=Reference address, Y=Result Descriptor, X bits 0-7=IO.M.EVENTS, X bits 8-23=IO.MCP.IO.
- @07@ GISMO requested control of the processor, returning a result that the Micro MCP is unable to handle (the T register contains the result returned).
- @08@ An I/O dispatch to GISMO returned a result that the dispatch was unsuccessful (the T register contains the result returned).
- @09@ A communicate message was not within the Base-to-Limit space of the program doing the communicate.
- @10@ The Micro MCP was requested to handle a communicate that it is not capable of handling.
- @11@ Irrecoverable error in the Micro MCP. The T register contains the first six digits of the Micro MCP source sequence number /where the halt occurred.
- @12@ Micro MCP CONDITIONAL.HALT. The T register contains the first six digits of the Micro MCP source sequence number where the halt occurred.
- @13@ On a delayed random I/O, the RESULT.DESCRIPTOR indicates that the I/O operation has been initiated, but the IO.MCP.IO field is in an improper state.
- @20@ The HI.PRI routine of the Micro MCP was given control, yet no high-priority interrupt was found in the Micro MCP INTERRUPT.QUEUE.
- @21@ A high-priority interrupt was found, but the IO.COMPLETE bit (bit 0 of the RESULT.DESCRIPTOR) was not set, indicating that the I/O was not yet complete.
- @22@ A dispatch of a POCKET.SELECT operation through GISMO returned a result indicating that the dispatch was unsuccessful (T=Result returned).
- @23@ A POCKET.COMPLETE or DOUBLE.DOCUMENT result was expected, but not obtained, from a POCKET.SELECT dispatch (T=Result returned).

MCP HALTS (L=@000011@)

When the MCP executes an explicit HALT instruction, the L REGISTER is set to @000011@ and a parameter is loaded into the T REGISTER to further define the nature of the halt. Usually, this parameter is the first six digits of a sequence number in the MCP itself; however, some halt conditions occur at more than one place in the MCP. These are given a common identifier, as follows:

T REGISTER	REASON
@111111@	No memory
@333333@	This MCP is missing patches. Push START to continue.
@444444@	HALT ODT input message was entered. Push START to continue.
@Cxxxxx@	
@Dxxxxx@	Disk I/O Error. Push START for Port, Channel, and Unit.
@Exxxxx@	
@Fxxxxx@	
@000Cxx@	Systems Software Compatibility Error The low-order eight bits of the T REGISTER contains two 4-bit numbers identifying the two programs that are incompatible, as follows:

- 1 System Initializer
- 2 GISMO
- 3 Micro MCP
- 4 MCP
- 5 SDL2 Interpreter

All other halts point to a specific sequence number in the MCP. The halts marked with an asterisk (*) can be pushed through by pushing START. All other halts are irrecoverable; a memory dump should be taken and submitted with supporting documentation and a B 1000 Field Trouble Report.

T Register	Reason for Halt
@037870@	Attempt to initiate in-process I/O.
@037925@	DISPATCH to invalid Port or Channel.
@038000@	QLOCK.COUNT overflow.
@038020@	QLOCK.COUNT underflow.
@038615@	Got an INTERRUPT from the Micro MCP.
@038630@	Memory parity on dispatch (Interrupt from Channel 15).
@039438@	MCP attempted a disk I/O, but the port and channel are not for a disk unit.
@039439@ *	A disk I/O did not complete in 30 seconds. Push START once - T contains the result descriptor. Push START again - T contains the port, channel, and unit. Pushing START again causes the MCP to wait for 30 seconds more.
@040475@	MCP asked to retry a disk operation that was not complete with an I/O exception.

T Register	Reason for Halt
@040581@ *	Bad memory statistics (Non-release MCP only).
@040708@	Broken memory link.
@040713@	Incorrect linkage found in queue structures, or an incorrect item type in queue available structure, or a bad backward link in queue available chain.
@041494@	Bad "pointer" field in memory link.
@043010@	Segment dictionary marked SAVE.
@043602@	A memory link which should have been marked SAVE was not so marked.
@044320@	Broken Memory Link.
@046045@	Could not open ODT queue.
@046060@	Could not find ODT_Q_FILE.
@046385@	Invalid interrupt for the MCP.
@046685@	Invalid interpreter-generated communicate.
@047741@	MCP fell out of the OUTER_LOOP.
@049169@	SYSTEM/ELOG disappeared from the system disk.
@049174@	Drive Transformation Table is incorrect.
@049545@ *	Integrity error on system disk found at clear/start.
@050755@	Failed to absolutize GISMO or SDL Interpreter.
@051035@	Cannot find an IOAT for this system disk.
@051656@ *	ODT failed to respond to a test op. Push START for ODT port and channel. Push START again for the reference address.
@051722@	Tried to CM and clear/start 12.0 software on an I/O system disk. <i>12 CSV too small</i>
@051744@	The Cold Start Variables have been <u>changed</u> by <u>this</u> MCP; please clear/start again. <i>line to COEDS/A04/174</i>
@052300@	Invalid dispatch while attempting to rewind a tape unit at clear/start.
@052592@	A control did not respond to TEST OP after a 30-second wait (at clear/start). Push START once - T contains port and channel. Push START again - T contains I/O Descriptor address.
@052640@	Too many datacomm port/channels on this system.
@052727@	Miscalculated required size of the IOAT during clear/start.
@054268@	ODT memory initialization problem at clear/start.
@054536@	Failed to find a system disk that clear/start had already found once before.
@055020@	Unexpected Port Channel Unit found for disk drive.
@058449@	Micro MCP segment fault improperly routed.
@062020@	Dispatch to invalid port or channel.
@062100@	Dispatch to invalid port or channel.
@062240@	MCP is lost during Punch Check Recovery.
@062760@	Dispatch to invalid Port or Channel.
@069470@	I/O Complete for ROLLOUT, but that task is not being rolled out.
@069740@	In IOCOMPLETE procedure, an I/O in question is not complete.
@070210@	Invalid IO.TYPE - 32.
@073042@	While looking for available space on a system disk, the MCP found the unit not to be a system disk.

T Register	Reason for Halt
@075950@	A dictionary is in memory, but its media bit is off.
@076790@	No memory for data overlays.
@078562@	DFH dictionary improperly formatted.
@078813@	DFH dictionary improperly formatted.
@079023@	DFH dictionary improperly formatted.
@080610@	Bad value passed to HEADER.UPDATE.
@081770@	Power Down MPF failed.
@085560@	An ISAM I/O was in process while updating buffers.
@088743@	MCP asked to handle file protection that should have been directed to the Micro MCP.
@088746@	MCP asked to handle file protection that should have been directed to the Micro MCP.
@090085@ *	MCP retrying an I/O for which there is no evidence of an error condition.
@090087@ *	MCP retrying an error for which there is no evidence of an error condition..
@091140@	An open pseudoreader file is not pointing at an existent pseudoreader.
@101020@	I.S. Fine table links are invalid.
@103680@	I.S. Currents missing for task.
@111111@	No memory.
@114060@	Incorrect interface between Q__DRIVER and Q__FILES.
@115200@	File not open condition not screened off by R__W__CALLER.
@115300@	Enhanced I/O not allowed.
@122512@	Illegal port in I/O descriptor.
@123481@	Tape control appears to be hung.
@126010@	Problem with skipping an empty area of a disk file.
@126090@	Call on GET.NEW.AREA with POSITION communicate is invalid.
@126480@	Could not update Base Pack header.
@126590@	Unrecognizable communicate called GET.NEW.AREA.
@140450@	Bad call on OPEN__QUEUE.
@140460@	Bad call on OPEN__QUEUE.
@140860@	Bad call on GET__QUEUE.MESSAGE.
@140870@	Bad call on GET__QUEUE__MESSAGE.
@140880@	Bad call on GET__QUEUE__MESSAGE.
@140890@	Bad call on GET__QUEUE.MESSAGE.
@140900@	Bad call on GET__QUEUE.MESSAGE.
@148390@	Tried to erroneously return non Q__disk to available.
@148477@	Problem returning Q__disk.
@153990@	Disk I/O error in usercode verification routines.
@154190@	No memory to verify security.
@154670@	Invalid usercode.
@158686@	Invalid parameter passed to usercode processing routines.
@158710@	Invalid usercode.
@159890@	Error in RIB list integrity.
@159975@	Bad call on GET__QUEUE__MESSAGE.
@159981@	Error in RIB list integrity.
@165430@	Cannot handle type INDIRECT in a Code Dictionary.

T Register	Reason for Halt
@173651@ *	During program initiation, attempted to bump the usercount in the interpreter file DFH - but the interpreter is already in the interpreter dictionary.
@173712@	During program initiation, attempted to assign a new interpreter to an invalid interpreter dictionary index.
@175110@	Lost track of number of users of a segment dictionary.
@176280@	PROG.SPAD.PTR not pointing to segment two of the PPB.
@179590@	No caller on an IPC Call.
@179805@	Parent job of a task has disappeared.
@181610@	Memory allocation for an RSN is incorrect in FIREPROG.
@181810@	Memory allocation for an ESN is incorrect in FIREPROG.
@188922@	RIB list corruption while handling a death_in_family
@192980@	Disk address in Task Variable Table equals zero.
@193284@	Lost track of a task's parent.
@193850@	No schedule entry for successor on unconditional execute.
@193903@	After the system disk had been squashed, insufficient memory to fixup addresses in the name table.
@195280@	Q_KEY stored in QPTR is now bad.
@195484@	Tried to return a chunk of disk whose address equals zero.
@195525@	An RSN had a zero task_variable_table address.
@198610@	Q_KEY for ODT_QUEUE is blown.
@200860@	TERM_FINAL failed.
@201920@	Terminating a task that does not exist on disk.
@202350@	Terminating a task that does not exist on disk.
@209186@	A page dictionary was illegally overlaid.
@210230@	Cannot find the file that we are currently using.
@210880@	SY_TYPE equals SYSTEM_DESC and SY_FILE on.
@211670@	Environment address destroyed in MOVE_ES_REMNANT.
@217980@	IO_MCP_IO on a user file in not equal to zero.
@220000@	In ROLLOUT for a frozen task.
@221710@	Could not find Pseudo Reader that was in use.
@226251@	Illegal parent job number for a task.
@226253@	Task points at another task's task_variable_table.
@227130@	No such task scheduled.
@227210@	Invalid key passed to CLOSE_QUEUE.
@227700@	Scheduled a task that does not exist on disk.
@232360@	Invalid MICR communicate.
@238570@	Discovered bad pack label.
@245120@	I/O will not complete while trying to ready a disk.
@245440@	No IOAT for current disk unit (MAKE_DISK_ABSOLUTE).
@248140@	Cannot find MPF_INFO_TABLE already in use.
@249050@	Bad disk file header.
@250080@	Cannot find entry in MPF_TABLE.
@250230@	Cannot find MPF_INFO_TABLE already in use.
@256290@	Tried to remove tracks from a non-disk device.
@256942@	Bad disk PCU found in temporary table management.

T Register	Reason for Halt
@257080@	No disk available for temporary available table.
@257232@	A PCU of zero was found while putting a chunk of disk into the temporary available table.
@258005@	A chunk of disk with a PCU of zero was found in an available table.
@258150@	MCP was asked to find the temporary available table on a non-disk PCU.
@258858@	Attempted to permanently allocate disk on a RAM unit.
@261102@	Tried to return disk to an offline disk device.
@262770@	Trying to return disk already in the available table.
@262880@	Returning an invalid disk address (MSG SPOUTED).
@263140@	Returning non zero disk address with zero PCU.
@263330@	Returning non zero disk address with zero PCU.
@264842@	No disk to extend an available table.
@269930@	Returning non zero disk address with zero PCU.
130 240283 @270290@ *	Invalid available table, addresses out of sequence. Push START to continue and run SYSTEM/PANDA to check disk.
@271820@	Bad entry found in temporary disk available table.
@289540@	Bad disk directory.
@290140@	Irrecoverable disk I/O error on file header read.
@291860@	No disk available for directory expansion.
@292020@	Irrecoverable disk I/O error on file header read.
@292260@	Disk directory has been corrupted.
@293952@ *	Problems while doing disk directory manipulation.
@296425@	SYSTEM/ODT remote file open denied - file missing.
@296430@	SYSTEM/ODT remote file open denied - file locked.
@296435@	SYSTEM/ODT remote file open denied - adapter missing/load failed.
@296440@	SYSTEM/ODT remote file open denied - MCS denied open.
@296445@	SYSTEM/ODT remote file open denied - too many remote files.
@296450@	SYSTEM/ODT remote file open denied - lsn missing.
@296455@	SYSTEM/ODT remote file open denied - too nested.
@296460@	Network controller run error detected.
@296465@	SYSTEM/ODT remote file open denied - too many stations.
@297605@	Insufficient system disk to open a queue file from the MCP to SYSTEM/ODT.
@297310@ *	Illegal parameter passed to procedure SPOUT__INVALID.
@298926@	Invalid disk type found in the IOAT.
@301065@	Missing LOG file.
@301087@	Parameters to SYSTEM/LOG have unexpectedly changed.
@301130@	No Disk for header for new log.
@301160@	Parameters to SYSTEM/LOG have unexpectedly changed.
@301470@	Missing LOG file.
@301481@	Parameters to SYSTEM/LOG have unexpectedly changed.
@301540@	Missing LOG file.
@301836@	Parameters to SYSTEM/LOG have unexpectedly changed.
@301860@	MCP tried to transfer the ODTLOG.
@302195@	Incorrect LOG__MIX__INFO for a task.

T Register	Reason for Halt
@302410@	Attempted to reallocate LOG_MIX_INFO for a task.
@303005@	No subport address for subport log entry.
@305665@	Failed to QUEUE a message for AUTOBACKUP.
@305715@	Failed Receiving Msg from QUEUE for AUTOBACKUP.
@305735@	Failed Receiving Msg from QUEUE for AUTOBACKUP.
@305740@	Failed receiving msg from QUEUE for AUTOBACKUP.
@305745@	Failed Receiving Msg from QUEUE for AUTOBACKUP.
@305760@	Failed closing queue for AUTOBACKUP.
@310760@	Port file is open but subport-attributes disk space is not assigned.
@313350@	Exceeded subport index while looking for change event.
@328980@	SYSTEM/DUMPFIL not large enough for DM command.
@332010@	A scheduled SORT vanished from the schedule.
@333333@ *	This MCP is missing patches.
@344907@	The program which called a SORT has disappeared.
@349240@	An I/O did not complete after a 30-second wait in the IO.ERROR routine. Push START once – T contains I/O descriptor address. Push START again – T shows whether OP was complete.
@357803@	Unexpected port/channel/unit found for disk drive
@384440@	Dispatch to an invalid port or channel.
@387880@	FIB indicates partial block but block count = 0.
@401510@	Tape I/O did not complete after a 30-second wait. Push START once – T contains I/O Descriptor address. Push START again – T shows result descriptor field.
@407742@ *	The PAN Name Table Entry is empty.
@408020@	Read of a disk pack label failed to complete in thirty seconds.
@411760@	Non-card device in CARD_STATUS.
@419600@	Tape I/O did not complete after a 30-second wait. Push START once – T contains I/O Descriptor address. Push START again – T shows result descriptor field.
@444444@ *	HALT ODT command was entered.
@445690@	Cannot find translate file that was already in use.
@452000@	No space allocated for ANSI.BUFFER.SPACE.
@477690@ *	Problem returning disk to available tables.
@478420@	Cannot find TRANSLATE file that was already in use.
@482926@ *	FIB and FPB for an open file differ in PROTECTION.
@483341@	Disk file header management problem..
@486720@	Missing file header for a multipack file.
@487610@	Cannot find pack that is already in use.
@493110@	Verify usercode failed when OPEN.SET.OVERRIDE.
@493571@	Mismanagement in processor scheduling.
@514910@	Cannot find translate file that was already in use.
@521700@	Pseudo Reader links are invalid.
@535560@	Queue not a queue or not a disk queue.
@535620@	Cannot find disk.
@536460@	Invalid parameters passed to OPEN_QUEUE.
@537010@	Invalid parameters passed to CLOSE_QUEUE.
@537920@	No multiline for remote open for SYSTEM/ODT.

T Register	Reason for Halt
@538070@	Invalid parameters passed to PUT_QUEUE_MESSAGE.
@540430@	Bad call on OPEN_QUEUE.
@543300@	Device is not a data recorder.
@545220@	Attempted to open a data recorder other than input or output.
@545720@	Attempted to open a data recorder other than input or output.
@548480@	Invalid parameters passed to PUT_QUEUE_MESSAGE.
@554620@	Invalid parameters passed to CLOSE_QUEUE.
@563630@	Could not power down Continuation Pack.
@563810@	Cannot find Base Pack.
@564710@	Lost a Disk File Header for a Pseudo Reader.
@564930@	MPF_TABLE has been destroyed.
@566822@	RETURN_DISK failure when closing a disk file.
@574664@	Queue management problem.
@575180@	Invalid parameters passed to PUT_QUEUE_MESSAGE.
@575330@	Invalid parameters passed to CLOSE_QUEUE.
@575360@	Invalid parameters passed to CLOSE_QUEUE.
@576120@	CLOSE_QUEUE invalid queue address.
@576200@	CLOSE_QUEUE invalid queue address.
@577280@	Invalid parameters passed to PUT_QUEUE_MESSAGE.
@577940@	Invalid parameter returned from PUT_QUEUE_MESSAGE.
@578000@	Invalid parameter passed to CLOSE_QUEUE.
@578020@	Invalid parameter passed to CLOSE_QUEUE.
@585270@	ISAM Subfile File Security violation.
@586690@	ISAM Subfile missing for HANG_PROGRAM_UP.
@592820@	ISAM Close Remove is missing a file.
@592970@ *	ISAM global file disappeared.
@593590@	Mismanagement of ISAM number of users.
@596230@	Begin address of I/O higher than MCP limit register.
@596280@	Dispatch to invalid Port or Channel.
@600670@	Failed to deliver an ENABLE_QUEUE message.
@601020@	PUT_QUEUE_MESSAGE failed or bad parameters passed.
@601550@	Failed to deliver a DISABLE_QUEUE message.
@602340@	PUT_QUEUE_MESSAGE failed.
@604100@	PUT_QUEUE_MESSAGE failed or bad parameters passed.
@604420@	PUT_QUEUE_MESSAGE failed or bad parameters passed.
@605170@	Invalid queue address passed to CLOSE_QUEUE.
@605230@	GET_QUEUE_MESSAGE failed - empty queue.
@605260@	GET_QUEUE_MESSAGE failed.
@605270@	GET_QUEUE_MESSAGE failed.
@605280@	GET_QUEUE_MESSAGE failed.
@605290@	GET_QUEUE_MESSAGE failed.
@606930@	Failed to deliver a queue message.
@607940@	PUT_QUEUE_MESSAGE bad parameters passed.
@608010@	RECEIVE did not RETURN.
@611600@	Could not find an LSN that should be present.
@615485@	Bad linkage in queue available buffer chain.
@616210@	Attempt clean-up of queue message not in-process.

T Register	Reason for Halt
@617320@	I/O Descriptor not complete.
@618050@	No message found.
@619830@	No message found.
@624770@	Queue memory link data structure broken.
@625060@	Queue must be empty here.
@625634@	RETURN__DISK problem while closing a queue file.
@625853@	RETURN__DISK problem while closing a queue file family.
@629034@	RETURN__DISK problem after opening a queue.
@629100@	Bad FIB or FPB address while opening a queue file family.
@629142@	Unable to successfully open a member of a queue file family.
@630390@	Queue FIB not in memory.
@631030@	Invalid return from CLOSE__QUEUE.
@631880@	Invalid parameters passed to OPEN__QUEUE.
@634500@	Attempted to handle enable/disable reply with no NDL handler.
@634530@	Attempted to handle enable/disable reply for a non-present remote FIB.
@635360@	No memory to add station.
@635980@	Queue close problems while firing a new controller.
@638500@	Invalid parameters passed to PUT__QUEUE__MESSAGE.
@639030@	Invalid parameters passed to PUT__QUEUE__MESSAGE.
@639690@	Invalid parameters passed to PUT__QUEUE__MESSAGE.
@640380@	Invalid value returned from PUT__QUEUE__MESSAGE.
@643920@	Unexpected return from PUT__QUEUE__MESSAGE.
@643930@	Unexpected return from PUT__QUEUE__MESSAGE.
@644630@	Bad call on OPEN__QUEUE.
@644650@	Bad call on OPEN__QUEUE.
@692560@	Port I/O request with no FIB.
@699370@	Invalid parameters passed to OPEN__QUEUE.
@699490@	Invalid parameters passed to OPEN__QUEUE.
@707430@	Invalid parameter returned from PUT__QUEUE__MESSAGE.
@707540@	BNALIO with no FIB.
@711800@	GET__QUEUE__MESSAGE problem.
@711820@	GET__QUEUE__MESSAGE problem.
@711830@	GET__QUEUE__MESSAGE problem.
@718732@	Parameter passed to SEARCH__EQUALS was not 30 characters in length.
@719186@	Parameter passed to SEARCH__EQUALS__TASK__SYNTAX was not 30 characters in length.
@724880@	Could not successfully enqueue an OBJ statement.
@724890@	Could not successfully enqueue an OBJ statement.
@727830@	IPC Caller no longer in the mix.
@744640@	Prior task no longer in the mix.
@754510@	Invalid parameter passed to CLOSE__QUEUE.
@754590@	Bad call on CLOSE__QUEUE.
@756875@	Failed receiving message from QUEUE for AB.
@756880@	Failed receiving message from QUEUE for AB.
@756905@	Failed closing queue for AB.
@757015@ *	MCP exited AB with AB = 0.
@758880@	Cannot find task that zipped "AT" message.

T Register	Reason for Halt
@759000@	FIND__UNIT failed to find a device.
@759645@	Incorrect call on OPEN__QUEUE.
@759650@	Incorrect call on OPEN__QUEUE.
@759960@	Incorrect call on PUT__QUEUE__MESSAGE.
@759965@	Incorrect call on PUT__QUEUE__MESSAGE.
@774291@	Disk error rate management problems.
@779132@	In IL, DMS environment running for a non-DMS job.
@779225@	RSN.RS__FILE exceeds number of files.
@779295@	RSN.RS__FILE exceeds number of files.
@790365@	Invalid USERCODE.
@808025@	Invalid USERCODE in PV.
@808065@	Invalid usercode in PV.
@808110@	Invalid usercode in PV.
@811931@	RF unable to read a file header into memory.
@815355@	Bad Pseudo Reader chain.
@820330@	No disk available for available table.
@827546@	WFL failed to be scheduled upon an MCP request to run.
@827558@	A command to WFL could not be successfully enqueued.
@834850@	ODT or ODX name table entry is empty.
@835245@	SEARCH__EQUALS failed in TH.
@852620@	Active task not in mix or schedule.
@854077@	Overflowed a task variable.
@856910@	Disk address in Task Variable Table equals zero.
@857000@	Task Number in Communicate does not match Task Number on disk.
@857420@	Disk address in Task Variable Table equals zero.
@872260@	Job Task not in mix.
@872480@	Disk address in Task Number Table equals zero.
@872510@	Query of active subtask.
@872690@	Task number in communicate does not match the task variable that was found by tasking routines.
@874530@	Disk address in Task Variable Table equals zero.
@876060@	Disk address in Task Variable Table equals zero.
@876090@	Task Number in Communicate does not match Task Number on disk.
@878456@ *	Bad title in WFL-generated data deck.
@880400@	WFL job had a null task__variable address
@907405@	When copying the code dictionary for the accessroutines of a DMS job closing a data base, an impossible condition was found.
@907420@	When writing the page dictionary for the accessroutines of a DMS job closing a data base, the job was found to be rolled out.
@907460@	When writing the page dictionary for the accessroutines of a DMS job closing a data base, corruption was found in the ESN or code dictionary.
@913080@	Zero port-channel on a DMS I/O.
@913100@	Bad unit hardware on a DMS I/O.
@913710@	Incomplete DMS write.
@913720@	Incomplete DMS write with exception.
@914520@	Incomplete DMS read.
@914530@	Incomplete DMS read with exception.

T Register	Reason for Halt
@916784@	When opening a DMS data base for its first user, there is no space in the audit buffer for the data base-open audit entry.
@917442@	An open or close of a DMS audit file was attempted for a non-DMS job.
@919290@	When opening a new area for a DMS audit file, the blocks-per-area count was greater than one.
@921132@	Although the DMS audit file status is OK, the audit file is not open.
@923490@	There has been a failure when copying the code dictionary for a DMS job closing a data base.
@923950@	When closing a DMS data base the DMS globals were missing from the DMS globals chain.
@924200@	There was a failure in closing the DMS environment of a job doing a data base close.
@924510@	When aborting a DMS job during data base open an impossible condition was encountered.
@924830@	No data base globals are allocated for an aborting DMS job with an incompletely open data base.
@926730@	Bad memory link encountered when releasing a DMS buffer.
@927363@	A DMS FATALERROR occurred on a data base that was audited to tape.
@927878@	When closing a DMS data base due to a FATALERROR, the disk audit file could not be closed.
@931140@ *	The DMS accessroutines gave a bad update-header communicate.
@941905@	A data inconsistency has occurred during DMS deadlock analysis.
@944761@	No user environment exists for a job opening a DMS data base.
@946250@	No disk file header exists in memory for an updated DMS datafile being closed.
@953580@	When binding DMS DASDL-generated code into the accessroutines at data base open, corruption was found in memory or the dictionary.
@972620@	Tried to switch to a non-existent environment.

Coldstart Tape Halts

When the COLDSTART/TAPE program halts, the L register is set to @000011@ and the T register contains the halt code. The following is a list of coldstart halt codes and their descriptions. On B 1990 systems, type GO and push the TRANSMIT key whenever the following instructions say to press the START button.

Halt Code	Description
@0C0001@	Disk I/O error. Press START button once to get result descriptor.
@0C0002@	Tape I/O error. Press START button once to get result descriptor.
@0C0003@	Unexpected data or result descriptor from Tape I/O.
@0C0004@	No tape control on system.
@0C0005@	No disk control on system.
@0C0006@	Disk not initialized in correct format.
@0C0007@	Trying to coldstart User pack.
@0C0008@	Missing SYSTEM tape. Make available and press START button.
@0C0009@	Missing required files from tape. Press START button until @0C0009@ is displayed again to obtain list of missing file numbers:

1 = MCPH
2 = SDL2/INTERP3M
3 = GISMO3
4 = SYSTEM/INIT
5 = MCPH/MICRO-MCP
6 = SYSTEM/COPY
7 = SDL2INTRIN/AGGREGATE
8 = SYSTEM/ODT
9 = SYSTEM/CONTROLLER
A = SYSTEM/PANDA
B = GISMO2

@0C000A@	Missing device on I/O dispatch.
@0C000B@	Insufficient disk for coldstart. Probably a bad Available or Master Available Table.
@0C000C@	Went beyond tape mark for a specific file. Try COLDSTART again. If it fails again, a corrupted SYSTEM tape is indicated.
@0C000D@	Missing tape mark.
@0C000E@	Invalid Tape HDR2 record.
@AAAAAA@	Cold start operation successfully completed. clear/start required.

Refer to the B 1000 Systems System Operation Guide, Volume 2, for a description of the COLDSTART/TAPE program and the COLDSTART/DISK program.

APPENDIX A

MCP MEMORY MANAGEMENT

The B 1000 computer systems were designed to cover a broad range of the computer market. In order to cover this range with a single operating system, it was necessary to implement virtual storage capabilities and apply the same techniques used for normal-state programs to the operating system itself.

The main memory requirements for any computer system are highly dependent on applications and operating procedures. This fact is even more true for a variable-length segment, virtual storage system such as the B 1000, which dynamically allocates memory to user programs as it is required. Such a system is able to keep many more programs in memory in order to provide higher processor utilization than are non-virtual systems or virtual machines with fixed page sizes (partitions). Since program segments are loaded only as needed, the memory requirements for programs on a machine such as the B 1000 (and the B 7000/B 6000/B 5000 systems as well) must be stated in terms of a working set, rather than either total program size or minimum memory required to run.

The working set for a program is that amount of memory that it most often needs during its execution to operate efficiently. This working set must also include the memory required for the functions requested of the operating system by the program, as well as certain operating system functions required for overall system control. The working set for the system as a whole is simply the sum of the working sets for all programs that are executed concurrently. If a program (or the system) is allocated less memory than its working set, it demands non-present segments at a rate that causes excessive segment overlying and reduced efficiency. When the performance of the system degrades appreciably due to memory restrictions, the phenomenon is known as thrashing.

CONCEPTS AND DEFINITIONS

Familiarity with certain basic concepts of memory management, including memory fragmentation, working set, and thrashing, is necessary for an understanding of the Memory Management System provided by the B 1000 MCP.

Memory Fragmentation

Memory fragmentation is the failure to allocate all of memory for useful purposes. Two varieties of memory fragmentation, internal and external, can occur. The type of memory fragmentation that occurs depends, respectively, on whether a system uses a paging or a segmentation mechanism for memory management.

In a paging system, all of memory is divided into equal-sized pages or partitions. Therefore, 100% of memory is assigned to usable pages, and external fragmentation does not occur. Since memory requests typically are of varying sizes, the last page assigned to a memory request is usually not full. This is internal fragmentation.

In a system based on segmentation, segment sizes are variable, so that only enough memory to satisfy a request is allocated for a program. Therefore, no internal fragmentation exists in such a system. Some memory is required as overhead for a memory link to describe each segment, however. A more serious problem is that an area of memory too small for use may become available between two segments of memory being used. This is external fragmentation.

Neither paging nor segmentation is clearly superior to the other. Each has its advantages and disadvantages. The primary advantage of paging is that it is easier for the operating system to handle. Segmentation, on the other hand, provides a much more reasonable structuring of memory, since only the space logically required for a given function is allocated to that request. Programmers need not be concerned with trying to structure their memory requirements into requests that are exact multiples of the page size of the system, allowing them to concentrate on completion of their primary tasks instead of having to think about unnecessary details. Segmentation does however, cause geography problems for the operating system, because external fragmentation checkerboards memory.

Burroughs has traditionally opted to use segmentation in its approach to memory management, and the B 1000 systems are no exception to this rule. Therefore, memory management on B 1000 systems is concerned with the algorithms and problems of segmentation.

Working Set

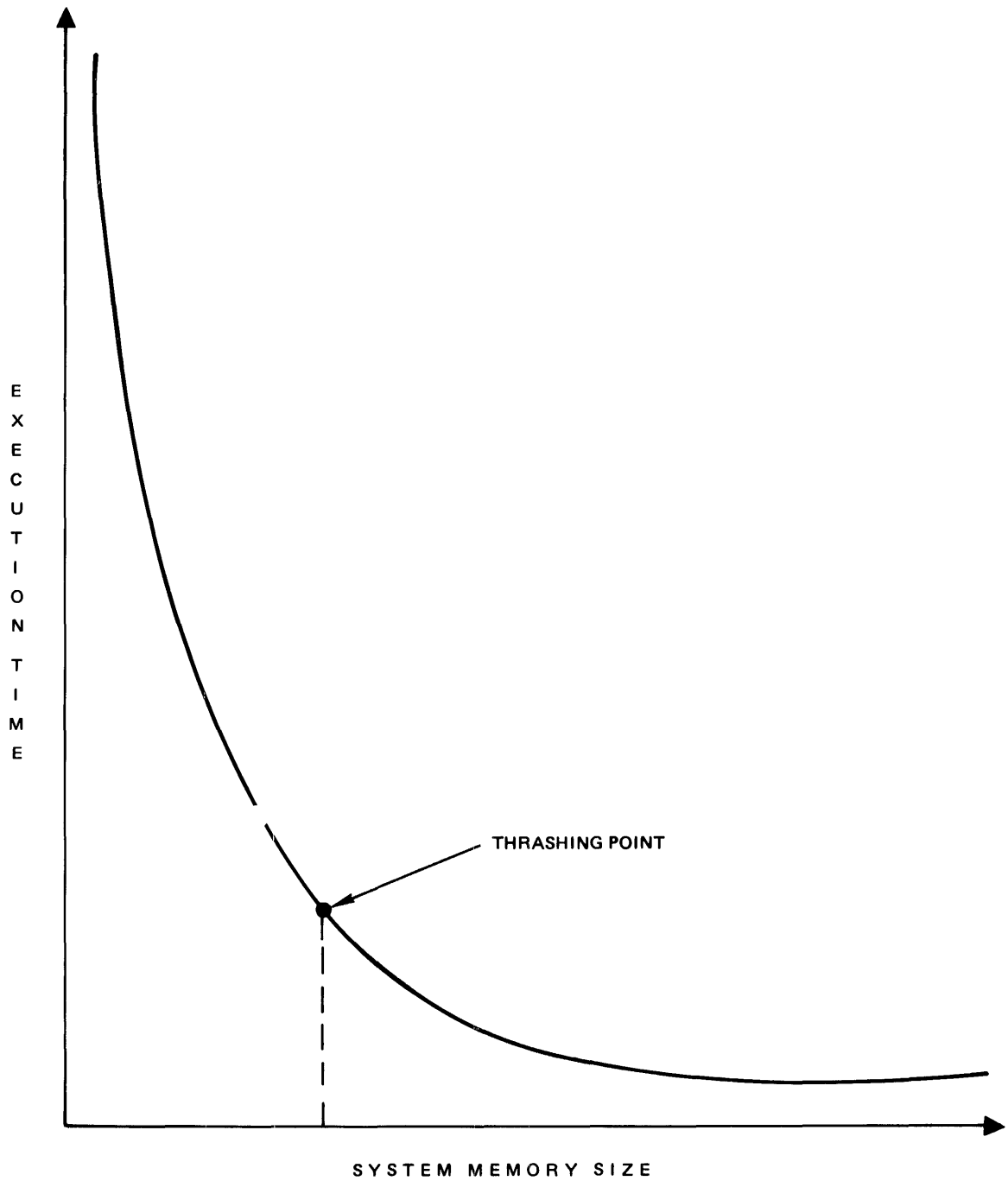
The term working set refers to the set of all program segments which are accessed during a specific time interval (of arbitrary length). The working set for a specific program is the set of data (the Run Structure), files, and code segments which it uses during such a time interval, plus the memory requirements of the operating system necessary to perform program-requested functions (READ, WRITE, OPEN, and CLOSE, for example). The working set for the entire system is the sum of the working sets of all currently active programs. The working set for a program, and especially for the system as a whole, can and often does change drastically over successive time intervals as tasks go from one phase of execution to another.

Thrashing

Thrashing is the condition that exists when the working set for a program or set of programs does not fit in real memory. Specifically, in order to bring in the next code segment for a program, the operating system has to overlay a currently active code segment. Then that segment has to be brought back in, and another active segment must be overlaid, and so forth.

One of the most serious problems confronting virtual storage systems is thrashing. As the amount of memory available for a constant programming task is reduced, the amount of degradation due to thrashing normally appears very gradual at first. As the available memory is further limited (by introducing additional programs into the system, opening files, requesting additional or larger code segments, and so forth), a point is reached where the degradation due to overlays increases rapidly. This is the point where the procedures in the working set no longer fit in memory and are competing for space. This point, referred to as the thrashing point, is shown graphically in figure A-1.

System performance suffers drastically when thrashing occurs. Throughput degradation of 100% and over is not unusual in such instances. In fact, in the worst case, absolutely nothing gets done except overlays.



G18609

Figure A-1. Thrashing Point

B 1000 MEMORY MANAGEMENT ALGORITHMS

No single memory management system is ideal for all situations. Consequently, the B 1000 operating system handles memory management with three separate levels of sophistication, using two different algorithms, in order to minimize the impact of more complex memory management schemes on those installations that do not need or want it. Installations that are satisfied with lower levels of the memory management system need not be concerned with the details of higher levels. This approach also allows users to ease into the more complex aspects of the memory management system smoothly, without being forced into an all-or-nothing decision.

Level One (First-In, First-Out)

The algorithm of Level One is basically a round robin (first-in, first-out) memory management scheme. When available memory space large enough to fulfill a request cannot be found by the operating system, one or more segments of in-use memory must be deallocated (overlayed). Overlayable memory is allocated starting from a left-off pointer which is then updated to point to the next lower segment in memory. Thus, the left-off pointer sweeps from high to low memory addresses until it reaches the first memory link, at which time it starts over again from the last memory link.

Each memory space has a touch bit that is set each time that memory space is accessed. As overlayable space is searched, if a touch bit is set, then the touch bit is reset and the search continues. If an overlayable space is found that does not have the touch bit set, then that space is used. The use of the touch bit allows code segments that are frequently used to stay in memory, overlaying those code segments that are not in use.

Save memory space, which cannot be reassigned until explicitly returned by the program to which it is assigned (for example, FIBs and file buffers), is allocated toward the high end of memory so that it will tend to be pushed together, thereby reducing the external fragmentation that such save space inherently creates.

Level One Advantages

1. External fragmentation of memory is minimized, since small available chunks of memory tend to be swept up and used as the left-off pointer sweeps through memory.
2. Although a simplistic decision about what segment to deallocate is made, this decision can be made quickly. This is a very important feature, because if enough memory is available to contain the working sets of the currently active programs, then the first priority of the memory management system is to get that working set in as quickly as possible.

Level One Disadvantages

1. The most serious flaw of Level One is that there is no straightforward method by which a system user or operator can determine when memory has been overcommitted, which causes thrashing.
2. The priority of a program using a segment is not considered when deciding to overlay that segment. Therefore, code segments of high-priority tasks are not protected from being overlayed by segments of lower-priority tasks. High-priority data communication tasks are prime examples of programs which often suffer because their segments are not protected from background tasks.

Level Two (First-In, First-Out With Thrashing Detection)

The second level of the memory management system implements detection for the thrashing condition. The same mechanism for determination of what segment to overlay (the victim selector) is used for Level Two as for Level One. Thrashing detection is invoked following the next clear/start operation by setting the THR option, when SYSTEM/INIT incorporates the thrashing detection code into GISMO. When GISMO, which is monitoring overlay activity, determines that thrashing is occurring and that it is not a temporary phenomenon, the operating system is notified. The operating system then performs the following two functions:

1. Stops more programs from being automatically started. This can be overridden by the system operator by using the PS system command to prod the schedule. Otherwise, the schedule is not automatically restarted until a program goes to end of job.
2. Sends the following message to the ODT:

SYSTEM IS THRASHING, SCHEDULE STOPPED

This message can be repeated either every time a program enters or leaves the MIX, or at every N.S-ECOND interval, as long as thrashing continues, depending upon the setting of the THRASH option of the MM system command.

When the system is shifting from one working set to another (as programs go to beginning of job or end of job, open or close files, or move from one phase of execution to another), memory is often overcommitted for a short period of time. This condition is acceptable if it does not persist for too long. One installation may, however, be willing to tolerate an overcommitment of memory for longer time intervals than another. For this reason, the THRASHING.SENSITIVITY option of the MM system command provides a means to adjust the sensitivity of the thrashing detection mechanism of the memory management system.

In addition, the maximum overlay rate that can be tolerated is highly dependent upon the speed of the disk from which the overlays are being done, since more overlays can be performed efficiently during a fixed time interval from fast disk than from a slow disk. For this reason the OVERLAY.RATE option of the MM system command provides a means to adjust this maximum allowable value.

The recommended value for the OVERLAY.RATE has been determined for the various disk types using their average access times (allowing for fixed operating system overhead required to obtain memory space and initiate the disk read) as shown in the following table:

Disk Type	Average Access	OVERLAY.RATE
B 9480 cartridge	80 ms	6
B 9481 cartridge	100 ms	5
B 9482 cartridge	55 ms	7
B 9484 pack	33.5 ms	10
B 9499 pack	42.5 ms	8
B 9371 head-per-track	20 ms	12
B 9371 head-per-track	40 ms	8
B 9470 head-per-track	5 ms	15

The default value for OVERLAY.RATE assigned by the operating system following a coldstart operation is 10.

Level Two Advantages

The advantage of Level Two is that system users and operators are aware of when their memory is overcommitted and are, therefore, able to do a much better task of maintaining a mix of programs which utilizes most of memory but does not cause thrashing to occur.

Level Two Disadvantages

The only disadvantage of Level Two is that approximately 140 more bytes of non-overlayable memory are required beyond that of the Level One mechanism.

Level Three (Memory Priority With Thrashing Detection)

Level Three of the memory management system includes the thrashing detection of Level Two, but has a different victim selector based on task priority and segment usage. The Priority Memory Management algorithm is invoked following the next clear/start operation by setting the MPRI option when the SYSTEM/INIT program incorporates the new victim selector code into GISMO.

In this level, requests for segments of memory are assigned priorities which are separate and distinct from processor usage priorities (refer to the MEMORY.PRIORITY control instruction attribute and the MP system command in section 2). No request for memory can cause a segment having a higher memory priority to be overlaid.

In a mix of programs having varying memory priorities, segments of high priority tasks actively in use are protected from segments of lower priority tasks. At fixed time intervals (known as the SAMPLING.INTERVAL) GISMO sweeps through all memory links on the system and examines a usage bit in each. This bit is set by the interpreter of the program when the code segment is accessed (that is, code in the segment is executed). If a segment has not been accessed since the previous sweep through memory, its priority is lowered by GISMO to the next lower memory priority active on the system. The segment is then protected at that priority for another SAMPLING.INTERVAL. If a segment is accessed at any time before being overlaid, it is restored to its original memory priority. In this way, segments of high-priority tasks are protected from those of low-priority tasks, and unused segments of any task tend to degrade to lower priorities and get overlaid.

In a flat mix (that is, a mix of programs having equal memory priorities), segments actively in use tend to stay in memory, while segments no longer being used tend to be overlaid. This cannot be made an absolute policy in a memory management scheme based on segmentation due to geography problems. For example, a very small inactive segment which has been allocated between two active segments may remain in memory longer than it otherwise would, because of its location. A flat mix has the additional advantage that it approaches the simplicity and efficiency of the Level Two algorithm as the system approaches thrashing.

The MCP sets the SAMPLING.INTERVAL value based upon the system memory size, as shown in the following table:

Memory Size	SAMPLING.INTERVAL Value
0- 261 KB	8 (0.8 seconds)
262- 523 KB	10 (1.0 seconds)
524- 785 KB	12 (1.2 seconds)
786-1047 KB	14 (1.4 seconds)
1048 KB	16 (1.6 seconds)

The SAMPLING.INTERVAL option of the MM system command provides a means to change the rate at which the GISMO sweeper is executed, although changes from the default value should not be necessary and are not recommended.

Level Three Advantages

1. Varying memory priorities protect active segments of higher-priority tasks from being overlaid by those of lower-priority tasks.
2. As in Level Two, the system operator is aware of when memory is overcommitted and can do a much better job of maintaining a mix of programs which utilizes most of memory but does not cause thrashing to occur.
3. Running with equal memory priorities tends to make the system run in a manner approaching that of Level Two. However, the added advantage is that unused segments degrade in priority and tend to be overlaid, while active segments tend to stay in memory.

Level Three Disadvantages

1. Approximately 170 more bytes of non-overlayable memory are required beyond that of the Level Two mechanism.
2. If tasks are run at varying memory priorities, external fragmentation of memory can be increased.

Extended Segment Decay

Level Three of the memory management system also allows protection of specified segments from overlay by lower-priority segments for an extended period of time (greater than the SAMPLING.INTERVAL) after they were last accessed. This capability is designed primarily to aid data communication installations which have no way of insuring that key segments of network controllers and other remote applications remain in memory. This problem can result in poor response time when low-priority batch or background tasks cause critical data communication program segments to be overlaid. It is not advisable to permit such important segments to be marked save (non-overlayable). However, Extended Segment Decay is only a little short of that capability.

There are two aspects to protecting key program segments:

1. Those segments which are to be protected for an extended period of time must be identified and marked. The means for accomplishing this is a utility program called SYSTEM/MARK-SEGS.
2. The length of time such segments are to be retained must be specified. This is done by setting the program attribute SECONDS.BEFORE.DECAY to some value between 0 and 600, inclusive (refer to the SECONDS.BEFORE.DECAY control instruction attribute in section 2).

The priority of segments which have been marked as important is not degraded until and unless those segments are not accessed for the number of seconds specified by the SECONDS.BEFORE.DECAY attribute. If SECONDS.BEFORE.DECAY is set to zero for a particular program, then all of its code segments (both those marked as important and unimportant) are treated as unimportant. Furthermore, SECONDS.BEFORE.DECAY is completely subservient to memory priority. A segment with a higher memory priority can overlay a segment with a lower priority no matter what the value of SECONDS.BEFORE.DECAY for the lower-priority task. SECONDS.BEFORE.DECAY only determines how long after a segment was last accessed it is able to retain a given priority.

Specifying a SECONDS.BEFORE.DECAY value for a program that has no segments marked as important by SYSTEM/MARK-SEGS has no effect.

Extended Segment Decay Advantages

Extended Segment Decay allows data communication users to guarantee that key segments of network controllers and other programs are not overlaid by lower-priority tasks for any fixed period of time (between 0 and 600 seconds) after they are last accessed.

Extended Segment Decay Disadvantages

Users of Extended Segment Decay can lock up more memory than they really need, and thereby degrade the performance of background tasks more than necessary.

FUNCTIONAL CHARACTERISTICS

The thrashing detection and priority memory management are the functional characteristics described in the following paragraphs.

Thrashing Detection

The following paragraphs describe the functional characteristics of thrashing detection used by the B 1000 operating system.

SAMPLING.INTERVAL

A value (in tenths of seconds) computed by the operating system from the B 1000 system memory size which specifies how often GISMO checks to determine whether thrashing is occurring. This value also specifies how often the GISMO sweeper is executed if the MPRI option is set.

OVERLAY.TARGET

The value (in number of overlays per SAMPLING.INTERVAL) computed by the operating system from the specified OVERLAY.RATE and the SAMPLING.INTERVAL.

MAX.SWEEP.INTERVAL

A value (in tenths of seconds) computed by the operating system from the THRASHING.SENSITIVITY specified, equal to one-third of the value of THRASHING.SENSITIVITY. If the MPRI option is set, MAX.SWEEP.INTERVAL specifies how often the sweeper is executed once GISMO determines that the OVERLAY.RATE has been exceeded.

OVERLAY.COUNTER

A count of the number of overlays that have occurred. The count is reset to zero at the end of each SAMPLING.INTERVAL.

SAMPLING.CLOCK

A field that is incremented at each TIMER INTERRUPT until it reaches the value of the SAMPLING.INTERVAL.

MEM.EXTEND.CLOCK

A field that is incremented by the SAMPLING.CLOCK at the end of each SAMPLING.INTERVAL (if the OVERLAY.COUNTER exceeds the OVERLAY.TARGET) until it reaches the value of the MAX.SWEEP.INTERVAL.

MEM.EXTEND.COUNT

A counter that is bumped each time the MEM.EXTEND.CLOCK exceeds the value of MAX.SWEEP.INTERVAL. If this counter reaches a value of 3, thrashing has continued for the length of time specified by THRASHING.SENSITIVITY, and GISMO notifies the MCP of this condition.

References to the sweeper are applicable only if the MPRI option is set (refer to the following paragraphs on Priority Memory Management).

Priority Memory Management

The Priority Memory Management mechanism, in addition to providing the thrashing detection capability described earlier, allows active code segments to be protected from overlay by lower-priority code. In order to prevent the total takeover of memory by high-priority code GISMO periodically sweeps through memory and lowers the priority of code segments which have not been accessed since the last time the sweep was performed. In this manner, segments not actively used by high-priority programs are reduced in priority (decayed) until they reach a point where they can be overlaid by lower-priority segments.

Figure A-2 presents the logic flowchart of the general operation of the thrashing detection code.

The logic flowchart presented in figure A-3 graphically depicts the process by which the sweeper in GISMO examines each memory link and decays the priorities of unused segments. Certain data names that have been used actually exist in the GISMO code. Other data names are fictitious, merely being used in the flowchart to represent a specific function. Their definitions are as follows:

DECAY.INTERVAL

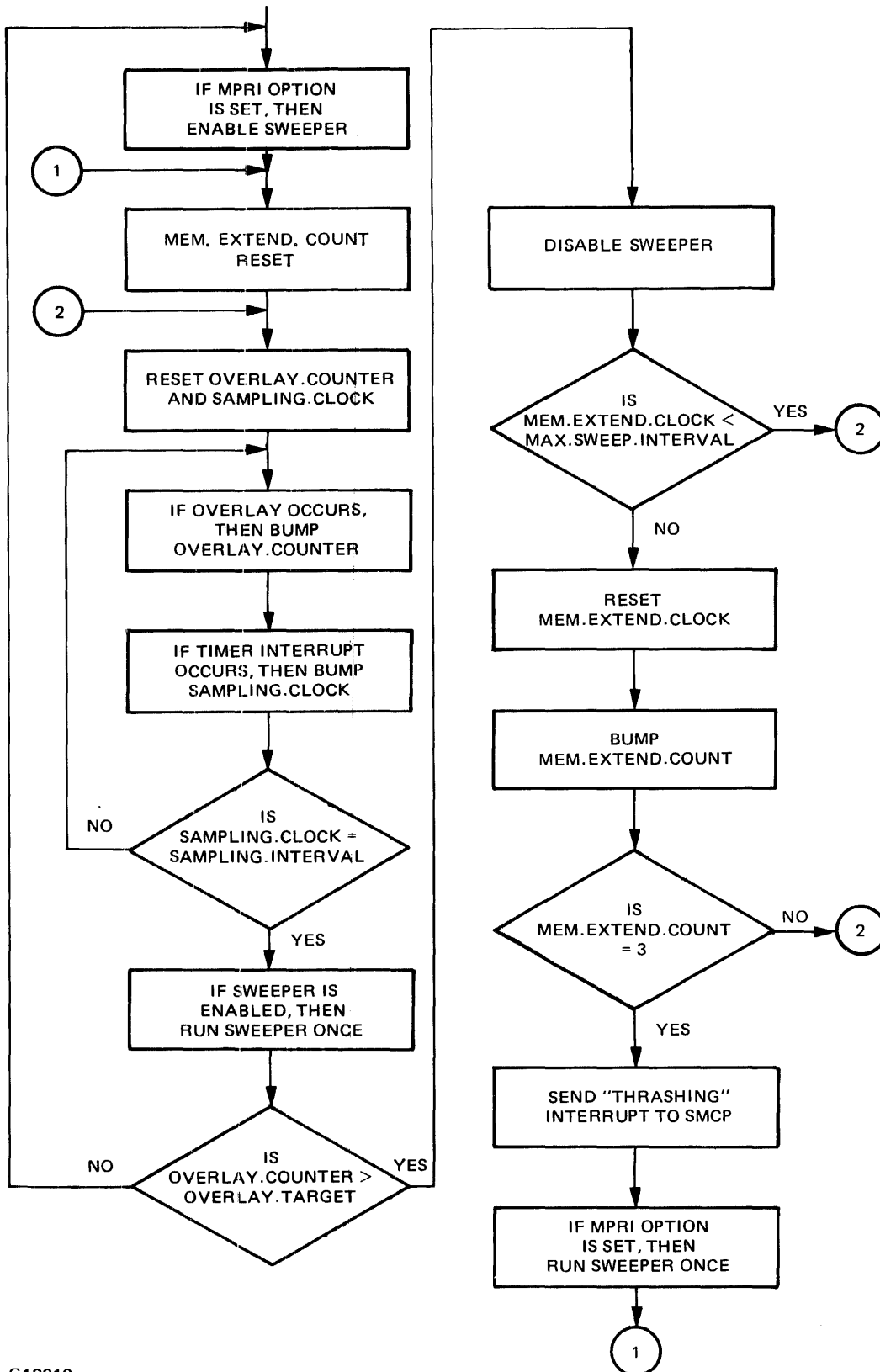
A value computed from the SAMPLING.INTERVAL and the SECONDS.BEFORE.DECAY specification which specifies the number of memory sweeps during which an unused segment cannot be reduced in priority. For example, if the SAMPLING.INTERVAL is 8 (0.8 seconds) and the SECONDS.BEFORE.DECAY attribute for a program is set to 20, the DECAY.INTERVAL for all important code segments is set to 25. In other words, the code segment is protected from decay for 25 sweeps through memory ($25 * .8 = 20$). Code segments which have not been marked as important are always marked with a DECAY.INTERVAL of 0.

MEMORY.PRIORITY

The value specified by the MEMORY.PRIORITY control instruction attribute or the MP system command.

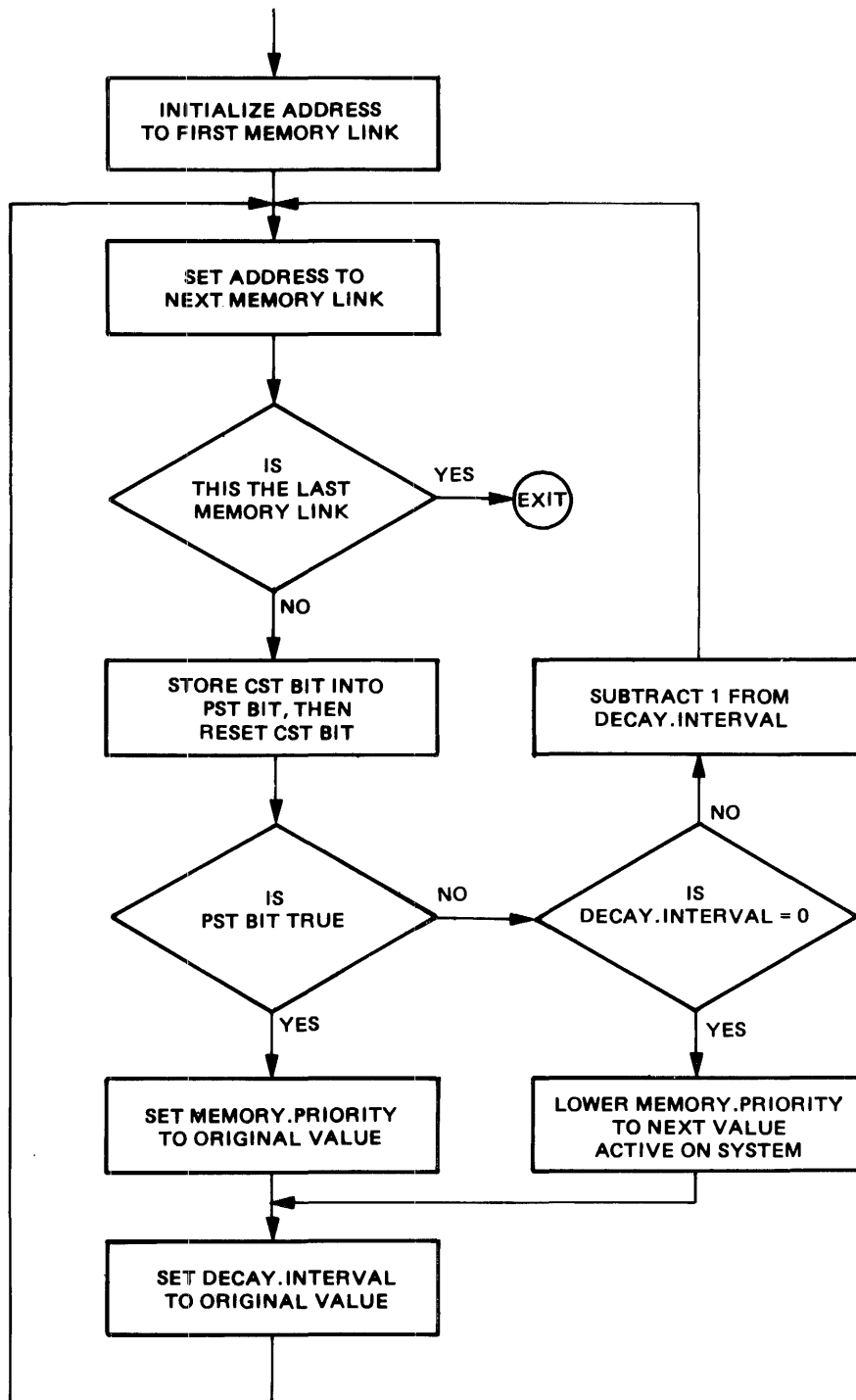
CST and PST

Two bits in the memory link adjacent to a segment of memory that indicate its in-use status. The CST (CURRENT.SCAN.TOUCH) bit is set by an interpreter whenever program control is passed to the adjacent code segment. The CST bit is reset only by the sweeper in GISMO. The PST (PREVIOUS.SCAN.TOUCH) bit contains the setting of the CST bit from the previous execution of the sweeper.



G18610

Figure A-2. Threshing Detection Logic Flowchart



G18611

Figure A-3. Memory Sweeper Logic Flowchart.

APPENDIX B

SYSTEM PERFORMANCE MONITORING

This appendix describes a feature which allows certain aspects of the B 1000 system performance to be dynamically monitored and displayed during execution using the system console.

GISMO contains code that can be optionally retained during the clear/start process to monitor system performance. The system operator is allowed to interact with this performance monitoring code through the system console, using the 24 data entry switches to specify the activity to be monitored, and reading the gathered information on the STATE light and the 24 main exchange lights on the B 1000 system.

Once familiar with the use of the performance monitoring system, it is possible for the system operator to dynamically tune the operation of individual programs and, in fact, the entire system to the workload being processed.

The use of the performance monitor function adds to system operations overhead and results in less than 5 percent degradation in overall system performance.

Through the console and the performance monitoring code in GISMO the system operator can monitor the following activities:

1. I/O activity by channel or by unit on a given channel.
2. The following subsets of processor activity:
 - 1) MICRO-MCP/MICRO.SCHEDULER time.
 - 2) SMCP time.
 - 3) User time (by mix number or total user time).
3. Overlay activity of the following;
 - 1) SMCP code, including interpreter and MICRO-MCP segments requested by the SMCP.
 - 2) User program code, including interpreter and MICRO-MCP segments requested by user programs (by mix number or total)
 - 3) User program data (by mix number or total).

SYSTEM CONSOLE

In terms of the performance monitoring operator interface, B 1000 systems have the same system console, consisting of 24 main exchange lights, 24 data entry switches (console switches), and the STATE light. The system operator enters monitor specifications through the console switches.

The 24 exchange lights are divided into two groups:

1. The left 16 lights are used as a bar graph.
2. The right 6 lights are used as a fixed display of various activities.
3. The 2 lights between the left 16 lights and the right 6 lights are not used.

The dual processor configuration of the B 1000 system has additional lamp capabilities. Any of the described functions can be monitored on the slave processor. Determining which processor is currently being viewed is controlled by the MASTER/SLAVE display knob or switch on the front console panel. Care must be taken as to what to monitor on the slave processor. Any monitoring of such work as I/O activity, overlays, and SMCP activity results in a null display by the lamps because none of this work is performed by the slave processor.

OPERATIONAL DETAILS

The following paragraphs describe the operation of the B 1000 performance monitoring capability.

Invoking The Performance Monitoring Capability

The performance monitoring lights are made active under the following conditions:

1. Appropriate MCP options must be set (using the SO system command).
2. Appropriate console switches must be set.
3. SE = 4 must be entered.

MCP Performance Monitoring Options

Performance monitoring requires retention of 2 to 4 discardable segments of GISMO by the SYSTEM/INIT program during clear/start. Which segments are retained is determined by the following three options:

FLMP

If this option is set, the GISMO code necessary to provide the fixed lamp display, is retained by SYSTEM/INIT during the clear/start procedure. There are no further options or specifications necessary for the fixed lamp display. Retaining the code at the time of the clear/start operation is all that is necessary to invoke the display.

VLCP

If this option is set, GISMO segments that allow specification and display of processor usage or overlay activity are retained.

VLIO

If this option is set, GISMO segments that allow specification and display of activity on the I/O subsystem, processor usage or overlay activity are retained.

Setting either the VLCP or VLIO options does not, in itself, cause any activities to be displayed; they only control which GISMO segments are to be retained during the clear/start process. When this option is set, the GISMO code necessary to provide the fixed lamp display, is retained by SYSTEM/INIT during the clear/start procedure. Since there are no further options or specifications necessary for the fixed lamp display, retaining the code at the time of the clear/start operation is all that is necessary to invoke the display.

The following warning is to be remembered in connection with the FLMP, VLCP, and VLIO options:

1. If a clear/start operation is performed using GISMO/DEBUG and any of the performance monitoring options is set, the STATE light (and the 24 main exchange lights for the B 1000 system) behaves quite differently than normal. This behavior may lead one to conclude that the system is malfunctioning when, in fact, it is running quite normally.

Specification Of Activities To Be Displayed

Specifying an activity to be displayed implies that whenever the requested activity is in process, the STATE light is ON, and whenever the requested activity is not in process, the STATE light is OFF.

Provided that the appropriate GISMO segments have been retained during the clear/start operation, the specification for what is to be displayed can be changed at any time. If the GISMO segments supporting the requested activity have not been retained, no activity is displayed.

The activity to be displayed is selected on the console switches from one of three groups:

1. I/O activity
2. Processor activity
3. Overlay activity

In order to make any selections, the console switches must first be enabled for use by GISMO, using the SE system command (SE 4; in section 2).

NOTE

In the discussions and figures that follow, a 1 indicates a switch or light that is ON, a 0 indicates a switch or light that is OFF, and an x indicates a switch or light position that does not apply.

Console switches 0 and 1 are always used to specify the B 1000 Bar Graph Scale Factor (refer to the paragraph entitled Bar Graph in this appendix). Console switches 2, 3, and 4 are used to specify which of the following three groups of activities to monitor and display:

Switch Value	Activity Group
000	Display disabled
1xx	I/O
01x	CPU
001	Overlay

The remainder of the switches have pre-defined functions that depend upon the Activity Group selected. These functions are discussed in detail in the paragraphs that follow.

Summary of Switch Specifications

Table B-1 summarizes the switch specifications used for requesting activities to be monitored and displayed.

Table B-1. Switch Specification Summary

Console Switch Number					
0 1 2 3	4 5 6 7	8 9 0 1	2 3 4 5	6 7 8 9	0 1 2 3
I/O	S	1 0	Channel Number	Tape Units (1-16, left to right)	
	a			Disk Units (0-15, left to right)	
CPU	l	1 C	U M I S	Mix Number (right-justified)	
OVLY	e	0 0	1 M U U C C D		

The meaning of the abbreviations used in table B-1 are as follows:

Abbreviation	Meaning
C	I/O control activity
U	User processor activity
I	Idle time
M	MICRO-MCP/MICRO.SCHEDULER processor activity
S	SMCP processor activity
MC	SMCP code, interpreter, and MMCP overlays
UC	User code, interpreter, and MMCP overlays
UD	User data overlays

Specification of I/O Activity Monitoring

Table B-2 depicts the switch specifications used to request the monitor and display activities on the I/O subsystem.

Table B-2. I/O Activity Switch Specifications

Console Switch Specification					
0 1 2 3	4 5 6 7	8 9 0 1	2 3 4 5	6 7 8 9	0 1 2 3
x x 1 0	Channel	Tape Units (1-16, left to right) Tape Units (0-15, left to right)			

Specification can be made (by turning on bit 2 of the console switches) that the STATE light is to be ON when there is an I/O operation in process on the selected channel, designated as a hexadecimal number (@0@-@E@) in bits 4-7 of the console switches. The meaning of the C bit (bit 3) and the units mask (bits 8-23) is dependent on the type of device selected described in the following paragraphs.

For devices other than disk and magnetic tape, the units mask is ignored. Selecting the channel and setting the C bit causes all activity on the specified channel to be shown. For example, the following console switch pattern causes all activity on channel 4 to be displayed (assume that channel 4 is not a disk or magnetic tape device):

xx11 0100 xxxx xxxx xxxx xxxx

For magnetic tape devices (including cassette) and disk devices not having overlapped seek (head-per-track disk and diskette), selecting the channel and setting the C bit causes all activity through the control to be monitored and displayed, regardless of the unit involved. If the C bit is reset, then only the activity for the units specified in the units mask (bits 8-23) is displayed. For example, the following switch pattern causes activity on units 2 and 6 of channel 10 (assume that these are magnetic tape units) to be displayed:

xx10 1010 0100 0100 0000 0000

In designating specific units on the unit mask, magnetic tape and cassette units are numbered from 1 to 16 (corresponding to tape units A through P), and disk units are numbered from 0 to 15 (corresponding to disk units A through P). The leftmost bit (bit 8) of the units mask represents unit A for the type of device selected, with the rest of the bits numbered accordingly.

For disk devices having overlapped seek capability (disk pack/cartridge), it is possible for the channel (control) to be idle when an I/O operation is in process; that is, when one or more units are seeking. If the C bit is set, the STATE light is ON only when the channel is busy (typically this excludes disk seek time). If the C bit is reset, the STATE light is ON whenever there is an I/O in progress for the units specified in the units mask (including disk seek time).

For exchanges, if the primary channel (the lowest channel on the exchange) is specified and the C bit is reset, all activity on the units specified in the units mask is shown, regardless of which control in the exchange actually performs the I/O operation. Setting the C bit causes only activity on the specified channel to be shown. If a secondary channel is designated, then the units mask is ignored (and no unit activity is shown); setting the C bit displays activity through the specified channel only. Thus, individual unit activity can only be monitored on an exchange by specifying the primary channel.

NOTE

TEST operations (including TEST.AND.WAIT operations) and PAUSE operations are never displayed.

Specification of Processor Activity Monitoring

Table B-3 depicts the switch specifications used to request the monitor and display activities on the processor.

Table B-3. Processor Activity Switch Specifications

Console Switch Number					
0 1 2 3	4 5 6 7	1 1 8 9 0 1	1 1 1 1 2 3 4 5	1 1 1 1 6 7 8 9	2 2 2 2 0 1 2 3
xx01	UMIS	Mix Number (right-justified)			

Specification can be made by turning off bit 2 and turning on bit 3 of the console switches so that the STATE light is to be ON during execution of:

1. User code
2. MICRO-MCP/MICRO.SCHEDULER code
3. Idle time
4. SMCP code

Any combination of these four classes of processor activity can be specified, using bits 4-7 of the console switches. If all four are specified together, the STATE light is always ON.

If the U bit (bit 4) is set and the mix number field (bits 8-23) is zero, then the STATE light is ON when any normal-state program is running. If the mix number field is non-zero, then the STATE light is ON only when the specified normal-state program is running. The mix number field is interpreted by GISMO as four 4-bit decimal digits, rather than as a 16-bit binary number. For example, the following switch pattern specifies that user processor activity for mix number 123 is to be displayed:

xx01 1000 0000 0001 0010 0011

As stated above, any combination of the four specifications can be used together. The mix number field is ignored, however, for any except the U bit. For example, the following switch pattern specifies that the STATE light is ON, whenever mix number 2345 is running or the processor is idle:

xx01 1010 0010 0011 0100 0101

NOTE

SOFT.IO time (in GISMO) is not isolated separately and is displayed as processor time used by whatever activity was running when SOFT.IO was invoked.

Specification of Overlay Activity Monitoring

Table B-4 depicts the switch specification used to request the monitor and display of code and data overlay activities.

Table B-4. Overlay Activity Switch Specifications

Console Switch Number					
0 1 2 3	4 5 6 7	1 1 8 9 0 1	1 1 1 1 2 3 4 5	1 1 1 1 6 7 8 9	2 2 2 2 0 1 2 3
x x 0 0	I M U U C C D	Mix Number (right-justified)			

Specification can be made (by turning off bits 2 and 3 and turning on bit 4 of the console switches) that the STATE light is to be ON when an overlay operation is in process; that is, from the time that the overlay request is recognized by the SMCP until the overlay is complete (including any disk operations) and the requesting program is ready to continue execution.

Following are the three types of overlays that can be shown:

1. MCP code overlays, including SMCP code segments, as well as SDL2 Interpreter and MICRO-MCP segments requested by the SMCP.
2. User code overlays, including user code segments, as well as interpreter and MICRO-MCP segments requested for a user program.
3. User data overlays.

When user code or user data overlays are requested and the mix number field is zero, then all user code or user data overlays, or both are shown. If the mix number field is zero, then only code or data overlays or both overlays for the selected user program are shown. For example, the following switch pattern specifies that SMCP overlays and user code overlays for all tasks in the mix are to be displayed:

```
xx00 1110 0000 0000 0000 0000
```

The following switch pattern specifies that SMCP overlays and user data overlays for mix number 5432 are to be displayed:

```
xx00 1101 0101 0100 0011 0010
```

NOTE

SDL, SDL2, UPL, or UPL2 overlays for PAGED arrays do not necessarily cause corresponding disk I/O operation, since the MCP attempts to use available memory outside of the Run Structure of the program for temporary storage of the array page, before resorting to disk storage.

Variable and Fixed Lamp Displays

The B 1000 system allows the 24 main exchange lights to be utilized as a bar graph and a fixed display of certain activities.

Bar Graph

The processor hardware timer is used by GISMO to accumulate the time involved in the selected activities. At each interval <n> (specified by the Bar Graph Scale Factor), the accumulated time is converted into a percentage of <n> and is shown on the Bar Graph. Each light in the Bar Graph group represents 1/16th of the interval <n> (left to right, 0 to 100%), with the following two restrictions:

1. If the activity or activities selected occurred during the interval <n> (regardless of how little), at least the leftmost light will be ON for the succeeding interval <n>.
2. If the activity or activities selected did not occupy 100% of the interval <n>, the rightmost light will not be ON for the succeeding interval <n>.

Bar Graph Scale Factor

The duration of the interval <n> is specified in the scale field of the console switches (bits 0-1), as shown in table B-5.

Table B-5. B 1000 Bar Graph Scale Factor

Switches	Interval <n>	Each Light Represents
00	400 ms	25 ms
01	800 ms	50 ms
10	1600 ms	100 ms
11	3200 ms	200 ms

As an example of what might be seen on the Bar Graph, assume that the following console switch pattern is selected:

0101 1000 0001 0010 0011 0100

This pattern specifies that user processor activity for mix number 1234 is to be displayed on the STATE light (and hence, the Bar Graph), and that the interval <n> is 800 milliseconds. Thus, every 800 milliseconds the leftmost 16 lights of the main exchange display is changed to show what percentage of the just-ended 800-millisecond interval that mix number 1234 was executing on the processor.

For example, if the main exchange lights are:

1111 0000 0000 0000 xxxx xxxx

then mix number 1234 was executing on the processor for 25% of the previous 800-millisecond interval (that is, 200 milliseconds). If the main exchange lights are:

1100 0000 0000 0000 xxxx xxxx

then mix number 1234 was executing on the processor for 12.5% of the previous 800 milliseconds (that is, 100 milliseconds).

As another example, assume that the following console switch pattern is selected:

0010 1001 1100 0000 0000 0000

This pattern specifies that I/O activity on channel 9 (assume that this is a disk pack channel), units 0 and 1, is to be displayed; the interval <n> is 400 milliseconds. If the main exchange lights are:

1111 1111 1111 0000 xxxx xxxx

then there was an I/O operation in process on either DPA or DPB for 75 percent of the previous 400-millisecond interval (that is, 300 milliseconds).

Fixed Lamp Display

The rightmost six lights of the main exchange display on the B 1000 processor form a fixed display group which show specific activities if the FLMP option is set. On these lights, six activities are shown exactly as they would appear, if selected, on the STATE light. The fixed light group is completely independent of whatever is being shown (if anything) on the STATE light and Bar Graph. There are no optional specifications; all that is necessary to invoke this display is to set the FLMP option and perform a clear/start operation (using GISMO), thus causing the necessary code to be retained by SYSTEM/INIT. None of the other two performance monitoring options (VLCP and VLIO) need to be set.

The six activities displayed are as follows:

Light	Activity
18	Any disk I/O operation in process. Here activity is shown for all disk channels and units on the system, whereas on the STATE light only one channel can be shown at a time.
19	Any overlay in process. The same as is shown on the STATE light when all three of the overlay options are selected and the mix number field is zero.
20	SMCP processor activity.
21	Idle time.
22	MICRO-MCP/MICRO.SCHEDULER processor activity.
23	User program processor activity.

Lights 20-23 are mutually exclusive. No two will be ON at the same time and together they indicate a total of 100% of processor usage.

APPENDIX C

DISK FILE ACCESS METHODS

The B 1000 MCP provides for three basic disk file access techniques: SERIAL, RANDOM, and DELAYED RANDOM. There are, however, a number of variations of these basic access techniques that are available for use. Some of these variations can be specified or declared within a source program, and all can be specified through attributes of the FILE control statement.

This appendix provides a general overview of the disk file access methods available, together with a description of their characteristics and restrictions.

Logical Versus Physical I/O

Normal-state programs perform logical I/O operations, consisting of READ and WRITE requests; the MCP converts these requests into the necessary data transfer operations and, where required, physical READ and WRITE operations. Logical I/O requests deal with records (sometimes referred to as logical records), whereas physical I/O operations deal with blocks (also called physical records).

Because programs deal only with logical records, the physical characteristics of a file (records-per-block, blocks-per-area, number of areas, and even the absolute disk address of each area, for example) are of no concern to the programmer. These physical characteristics are used only by the operating system, which transforms logical I/O requests from programs into the necessary physical I/O terms.

Any logical read operation by a program results in a transfer of data (the logical record) from the physical record buffer to a logical record work area allocated within the data space of the program. Inversely, a logical WRITE by a program results in a transfer of data (the logical record) from the logical record work area within the program's data space to the physical record buffer.

File Information Block

When any file is opened by a program, the SMCP allocates memory outside the data space of the program for use in managing the physical I/O of that file. This non-overlayable memory space contains the File Information Block (FIB), a structure which maintains information about the logical characteristics of the file as declared within the program, including logical record size, physical record size, records-per-block, current buffer pointer, logical record pointer, record and block count, and so forth. Since a file can be opened having logical characteristics that are different from its physical characteristics, the information in the FIB may be entirely different from information in the disk file header.

I/O Descriptors and File Buffers

For each buffer declared for the file, one I/O descriptor and memory space for the block is also allocated. This space is adjacent to the FIB, and is part of the non-overlayable memory space allocated for managing the physical I/O of the file. The I/O descriptor is used by the MCP for performing the physical I/O operations to or from the associated buffer.

Buffers are linked together in such a fashion that the next buffer to be used for a physical I/O operation is always the one that was accessed least recently. For SERIAL files, this means that the buffers are linked in a round-robin order. For RANDOM and DELAYED RANDOM files, the buffer links are dynamically updated by the MMCP to maintain them in order from the least to the most recently accessed.

Disk File Header

A disk file header (DFH) is associated with every disk file opened by a program. The DFH is a structure separate from the FIB which maintains the physical characteristics of a disk file, including logical record size, physical record size, records per block, blocks per area, areas in use, maximum areas allowed, end-of-file pointer, and the absolute disk address of each area assigned to the file. Also, since more than one program may have the same disk file open simultaneously, the number of users (or user counts) are also maintained in the DFH.

The disk file header for an old file (that is, one which has been entered in the disk directory) is stored on disk. When such a file is opened by a program, the SMCP reads the DFH into memory and uses the information stored in it to construct portions of the FIB.

The disk file header for a new file (that is, one which has not yet been entered in the disk directory) is constructed in memory by the SMCP when the file is opened. Temporary disk space for the DFH is obtained by the SMCP; this disk space becomes the permanent location of the DFH if the file is closed and entered into the disk directory.

The DFH may or may not be resident in memory while the associated file is open. If any programs which are sharing the file have the file opened `RANDOM` or `DELAYED RANDOM`, the DFH is always memory-resident; otherwise, the DFH is only brought into memory by the SMCP when information must be obtained from or stored into it.

DISK FILE ACCESS CHARACTERISTICS

The access characteristics of a disk file are basically determined by the method in which it is processed: `SERIAL`, `RANDOM`, or `DELAYED RANDOM`.

Serial Files

Records in a Serial disk file are generally obtained from or placed into the file in a sequential manner. That is, a logical record is made available from the next position in the file on a read operation, or a logical record is placed into the next position in the file on a write operation. The specific access characteristics of a serial file are dependent, however, on the way in which the file is opened (`INPUT`, `OUTPUT`, or `INPUT/OUTPUT`).

Input Serial Files

Serial disk files opened `INPUT` have the following characteristics:

1. When the file is opened, the SMCP initiates a physical read operation to fill each buffer assigned to the file.
2. The record address is updated by the MMCP to point to the next logical record to access following a logical read operation.
3. If a logical read operation empties the current buffer, the MMCP initiates a physical read operation to refill the buffer and updates the buffer pointer to point to the next buffer to access.
4. If the current disk area has been emptied, the SMCP obtains the address of the next disk area from the DFH just prior to initiating the physical read operation. If the next disk area is not allocated (that is, the area address in the DFH is zero), the SMCP positions the disk area pointer to the beginning of the next allocated disk area. This operation is completely transparent to the program.
5. Logical read requests beyond end of file are not allowed, and cause termination of the program if no EOF action is specified (for example, `AT END` or `ON EOF`).
6. When the file is closed, no physical I/O operations are initiated by the SMCP for the buffers assigned to the file.

Output Serial Files

Serial disk files opened OUTPUT have the following characteristics:

1. When the file is opened, no physical I/O operations are initiated by the SMCP for the buffers assigned to the file.
2. The record address is updated by the MMCP to point to the next logical record position to access following a logical write operation.
3. If a logical write operation fills the current buffer, the MMCP initiates a physical write operation to write the buffer to disk and updates the buffer pointer to point to the next buffer to access.
4. If the current disk area has been filled, the SMCP obtains disk space for the next disk area just prior to initiating the physical write operation, and stores the address of the new disk area into the DFH.
5. Logical write requests beyond end of the file are allowed, adding logical records to the end of the file (up to the declared maximum file size). The EOF.POINTER field in the FIB is updated by the MMCP to include the logical records that are added.
6. Logical write requests beyond the declared maximum file size are not allowed, and cause termination of the program if no EOF action is specified (for example, AT END or ON EOF).
7. If the current buffer is partially filled when the file is closed, the SMCP writes the partial block to disk. If the EOF.POINTER in the FIB is greater than the EOF.POINTER field in the DFH, the DFH is updated to reflect the new value for end of the file.

Input/Output Serial Files

The access characteristics of any serial disk file opened INPUT/OUTPUT (usually referred to as I/O Sequential) are determined solely by the setting of the EXTEND attribute for that file.

EXTEND Attribute Set

When the EXTEND attribute is set on an I/O sequential disk file, the logical I/O characteristics are as follows:

1. When the file is opened, the SMCP initiates a physical read operation for each buffer assigned to the file.
2. The record address is updated by the MMCP to point to the next logical record to access following a logical write operation.
3. The record address is updated by the MMCP to point to the next logical record to access prior to a logical read operation, but only if the previous logical I/O request was also a read operation.
4. Logical write requests beyond the end-of-file record are allowed, adding logical records to the end of the file (up to the declared maximum file size). The EOF.POINTER field in the FIB is updated by the MMCP to include the logical records that are added.
5. Logical write requests beyond the declared maximum file size are not allowed, and cause termination of the program if no EOF action is specified (for example, AT END or ON EOF).
6. If a logical write operation fills the current buffer, the MMCP initiates a physical write operation to write the buffer to disk and updates the buffer pointer to point to the next buffer to access.
7. If the current disk area has been emptied, the SMCP obtains the address of the next disk area from the DFH just prior to initiating the physical read operation. If the next disk area is unallocated (that is, the area address in the DFH is zero), the SMCP positions the disk area pointer to the beginning of the next allocated disk area. This operation is completely transparent to the program.

8. If the current disk area has been filled by write requests after reaching the end-of-file record, the SMCP obtains disk space for the next disk area just prior to initiating the physical write operation, and stores the address of the new disk area into the DFH.
9. If the current buffer has been updated, the SMCP writes the block to disk when the file is closed. If the value of the EOF.POINTER field in the FIB is greater than the value of the EOF.POINTER field in the DFH, the DFH is updated to reflect the new value for end of file.

EXTEND Attribute Reset

When the EXTEND attribute is reset on an I/O Sequential disk file, the logical I/O characteristics are as follows:

1. When the file is opened, the SMCP initiates a physical read operation for each buffer assigned to the file.
2. The record address is updated by the MMCP to point to the next logical record to access prior to a logical read operation.
3. The record address is not updated on a logical write operation, allowing the same record to be written more than once between logical read operations.
4. Since a logical write operation does not update the record address and a logical read operation beyond the end-of-file record is not allowed, records cannot be added to the end of the file. Logical write requests after the end-of-file record has been reached overwrite the last logical record in the file.
5. If a logical write operation fills the current buffer, the MMCP initiates a physical write operation to write the buffer to disk and updates the buffer pointer to point to the next buffer to access.
6. If the current disk area has been emptied, the SMCP obtains the address of the next disk area from the DFH just prior to initiating the physical read operation. If the next disk area is not allocated (that is, the area address in the DFH is zero), the SMCP positions the disk area pointer to the beginning of the next allocated disk area. This operation is completely transparent to the program.
7. If the current buffer has been updated, the SMCP writes the block to disk when the file is closed.

NOTE

The setting of the EXTEND attribute only has significance in connection with serial disk files opened INPUT/OUTPUT. The EXTEND attribute is ignored by the MCP for all other files.

Random Files

Records in a random file are generally obtained from or placed into the file in a random manner, based solely on the relative record number (the key) specified in the logical read or write request. That is, the logical record identified by the key is made available from the file on a read operation, or a logical record is placed into the file at the position specified by the key on a write operation.

Any logical read or write request by a program must specify the relative record number desired. The MMCP uses the relative record number to compute the physical position (disk area, block number, and record position within the block) of the logical record within the file.

The characteristics of random disk files are as follows:

1. When the file is opened, no physical I/O operations are initiated by the SMCP.
2. For files opened INPUT or INPUT/OUTPUT, if the block containing the requested logical record is not already in memory, the MMCP first initiates a physical read operation to read the block into the next buffer. Additional logical read requests do not cause a physical read operation, as long as the block remains in memory.
3. Logical read requests referencing an unallocated disk area or beyond the end-of-file record are not allowed, and cause termination of the program if no INVALID KEY action is specified.
4. Every logical write request causes the MMCP to initiate a physical write operation to write the block to disk.
5. Logical write requests beyond the end-of-file record are allowed, and cause the value of the EOF.POINTER field in the FIB to be updated by the MMCP to reflect the maximum allowable key.
6. A logical write operation to an unallocated disk area causes the SMCP to obtain disk space for that area just prior to initiating the physical write operation. The address of the new disk area is stored into the DFH.
7. When the file is closed, no physical I/O operations are initiated by the SMCP for the buffers assigned to the file. If the value of the EOF.POINTER field in the FIB is greater than the value of the EOF.POINTER field in the DFH, the DFH is updated to reflect the new value for end of file.

Delayed Random Files

The delayed-random access method is intended for use where random, input/output capability is desired but where, for the most part, the file is accessed sequentially.

The access characteristics of delayed-random files are identical to those of random files, with the following exceptions:

1. Logical read and write requests do not cause any physical I/O operations as long as the block remains in memory. A block in memory is overlaid if a request is made for another block not currently in memory. If the block chosen to overlay (the least recently accessed buffer) has been updated in memory by a logical WRITE operation, it is written back to its location on disk by the MMCP before the new block is read. Periodically (at each SMCP N.SECOND interval), all blocks that have been updated in memory are written to disk by the SMCP.
2. Since none of the physical I/O operations are overlapped with user processing on a delayed-random file, a file that is accessed more randomly than sequentially performs better as a random file than as a delayed-random file. Similarly, on a file accessed mostly sequentially, but where all physical I/O time is overlapped with user processing, a random file also performs better than a delayed-random file, because the program is not forced to wait for the physical write operations to complete.
3. Unblocked random files perform much better than unblocked delayed-random files. Because of this, the SMCP automatically changes any unblocked delayed-random file to random during file open.
4. When the file is closed, the SMCP writes back to disk any buffer in memory that has been updated. Note that the integrity of a delayed-random file can be guaranteed only if the file is closed. If the system is halted before the file is closed, integrity is guaranteed only if the last logical write operation occurred prior to the last SMCP N.SECOND interval; otherwise, buffers updated in memory will not have been written to disk.

AUDITED AND PROTECTED FILES

In the event of a system interrupt, the file attributes AUDITED and PROTECTED provide two methods of ensuring the integrity of an open file. These file attributes are described in section 4 of this manual and a discussion of the two protection methods follows.

The differences between the two attributes are:

1. The AUDITED file attribute preserves disk writes to the file by suspending a program until the write is complete.
2. The PROTECTED file attribute preserves the end of file (EOF) pointer by writing a protection pattern or updating the the EOF pointer after each write.

A file can be PROTECTED and AUDITED. In this case the data that has been written to disk and recovery of the EOF pointer are guaranteed.

NOTE

For indexed organizational files, the two attributes are the same because they were both implemented for indexed files before protected files were implemented.

AUDITED FILES

The user of an AUDITED file is guaranteed that a write to the file has taken place and that the data is on disk. If the file is an old file, that file will be on disk after a clear start or an abnormal program termination. If the file is a new file the file may be lost after a system halt; additions before the EOF pointer can not be guaranteed because the EOF pointer may not have been updated.

PROTECTED FILES

The user of a PROTECTED file is guaranteed that data written to disk before the end of file (EOF) pointer is recoverable after a clear/start operation or program termination. For serial access organized PROTECTED files the EOF pointer is adjusted to include the last record in the file after a clear/start operation is performed. For Random access sequential organized PROTECTED files, the EOF pointer is updated, if necessary, after each write.

When a serial file is extended the remainder of the last block and the next block are initialized with a protection pattern. When these write operations are complete, the disk file header (DFH) is updated to record that a protected file is being extended. As the user's block becomes full, a write operation of both the user's data and the next pattern block is initiated. A pattern write operation occurs two blocks ahead of the user's data block, and thus the program does not need to wait for the pattern write operation to complete. This process continues until the user program stops extending the file. The process is modified slightly at the end of an area. Then, after a clear start is performed, the end of file pointer can be adjusted by searching the file from the old EOF pointer until a protection pattern is found.

Sequentially organized files are recovered according to their physical file descriptions in the DFH. If a file is opened with a different record length or blocking factor than its physical description, data may be lost. Also, partial data transfers can occur if the halt button is pushed between transfers of data to the disk I/O control. It is recommended that the system be halted with the HALT ODT command or the interrupt switch because the GISMO program waits for disk input/output operations (I/O) to complete.

A random file is protected by updating its EOF pointer after each write operation. If the DFH is updated, then it is written to disk after the user's data block is written. Updating the DFH causes one extra random I/O per block. Since there is a write for each record in a block, a halt occurring between writing a block and updating the DFH, could cause, at most, one record to be lost. If the AUDITED file attribute is set, the program is suspended waiting for both the data write and DFH update. In this case, no records are lost.

There is extra I/O activity involved in providing the protection requested. Extra I/O operations are included in the block count for the file, and a count of the number of times the user program had to wait for a protection write is also maintained. Both of these counts are logged.

DISK FILE ACCESS SPECIFICATIONS

The access characteristics to be used by a disk file can be specified either in the source program or, following compilation, with attributes of the FILE control statement.

Source Program Specifications

Each source language available on the B 1000 system has different capabilities and limitations in regard to the specification of file attributes, and therefore, in the disk file access techniques that can be requested. Table C-1 summarizes the disk file access techniques that can be specified or changed using the FILE control statement following compilation.

Table C-1. Disk File Access Specification Summary

	BASIC	COBOL	FORTRAN	RPG	SDL/UPL
SERIAL FILES:					
INPUT	N	S	S	S	S
OUTPUT	N	S	S	S	S
INPUT/OUTPUT					
EXTEND	N	S	F	N	S
NO EXTEND	N	F	F	S	S
RANDOM FILES					
INPUT	N	S	F	S	S
OUTPUT	N	S	F	S	S
INPUT/OUTPUT	S	S	S	S	S
DELAYED RANDOM FILES:					
INPUT	N	F	F	F	S
OUTPUT	N	F	F	F	S
INPUT/OUTPUT	F	F	F	F	S
Legend:					
S Access technique can be declared directly in the source program.					
F Access technique cannot be declared directly in the source program, but can be specified through FILE attributes.					
N Access technique can be specified through FILE attributes, but is not compatible with the file access characteristics of the language.					

BASIC File Declarations

Disk files in the BASIC language are declared using the FILES statement. In order to facilitate the file handling characteristics of the BASIC language, all disk files are implicitly specified as random INPUT/OUTPUT by the compiler. Changes to this default, other than possibly changing random to delayed-random, are not recommended, and may cause undesired results during execution.

COBOL File Declarations

The COBOL language provides the ability to specify two access techniques for disk files, random and serial. These access methods are declared in the INPUT-OUTPUT SECTION of the ENVIRONMENT DIVISION, using the ACCESS MODE clause. The syntax is as follows:

```
ENVIRONMENT DIVISION.  
.  
.  
.  
.  
INPUT-OUTPUT SECTION.  
FILE-CONTROL.  
    SELECT file-name ASSIGN TO DISK  
        ACCESS MODE IS access-mode.
```

The access mode can be specified as either random or sequential. Delayed-random cannot be declared in a COBOL source program, but can be specified following compilation using the FILE control statement.

For all files declared as SEQUENTIAL, the COBOL compiler implicitly sets the the EXTEND attribute. This allows records to be added to the end of the serial disk file that is opened INPUT/OUTPUT. There is no means for specifying NO EXTEND in a COBOL source program; however, the FILE control statement may be used following compilation to reset the EXTEND attribute, if necessary.

FORTRAN File Declarations

The FORTRAN language allows disk files to be declared as either serial or random using the FILE declaration statement. Unless the key word random is included in the attribute list, the access mode is assumed to be serial.

Example:

```
Serial file: FILE 4 = INPUT, UNIT = DISK  
Random file: FILE 3 = SOURCE, UNIT = DISKPACK, RANDOM
```

Since all file opens are implicitly performed when the first logical read or write request is received by the MCP, the access technique for serial files is specified by the first logical I/O request. A read operation causes the file to be opened INPUT, and a write operation causes the file to be opened OUTPUT. Serial INPUT/OUTPUT files cannot be declared directly in the FORTRAN source program, but can be specified using the INPUT and OUTPUT attributes of the FILE control statement following compilation.

All random file in the FORTRAN language are implicitly opened INPUT/OUTPUT, and no syntax exists for declaring delayed-random files directly in a source program. Any of these default attribute restrictions can be explicitly changed through the FILE control statement by using the appropriate attribute specifications.

SDL/UPL File Declarations

The SDL and UPL compilers provide the ability to specify all allowable disk file access techniques, using the FILE declaration statement. The syntax is as follows:

```
FILE <file-name> (DEVICE = DISK <access> ...);
```

The <access> may be specified as any of the following:

Access	Access Technique
Serial	SERIAL (EXTEND attribute set)
Serial with Overwrite	SERIAL (EXTEND attribute reset)
Random	RANDOM
Delayed Random	DELAYED RANDOM

Files can be opened in the SDL and UPL languages either explicitly by using the OPEN statement, or implicitly when the first logical read or write request is received by the MCP. The attributes for an implied open can be specified as part of the FILE declaration statement in the source program, or can be modified following compilation using the FILE control statement.

RPG File Declarations

The access technique used by a disk file in an RPG program is declared in the File Description Specification for that file. The specific access is determined by the entries in columns 15, 16, 32, and 66, as shown in table C-2.

Table C-2. RPG Disk File Access Technique Specifications

Column				Attribute Set				
15	16	32	66	SERIAL	RANDOM	INPUT	OUTPUT	NEW
I	P			X		X		
I	S			X		X		
I	T			X		X		
I	D			X		X		
I	C				X	X		
I	P	I			X	X		
I	S	I			X	X		
I	D	I			X	X		
I	C	I			X	X		
I	P	I	A		X	X	X	
I	S	I	A		X	X	X	
I	D	I	A		X	X	X	
I	C	I	A		X	X	X	
O			A	X			X	X
O			A		X		X	
O		I	U	X			X	X
O		I	A		X		X	
O		I			X		X	
O		I			X		X	
U	C			X		X	X	
U	S			X		X	X	
U	D				X	X	X	
U	C	I			X	X	X	
U	S	I	A		X	X	X	
U	D	I	A		X	X	X	
U	C	I	A		X	X	X	

Refer to the B 1000 Systems Report Program Generator (RPG) Language Manual for detailed information about the entries shown above in table C-1 in columns 15, 16, 32, and 66.

Delayed-random disk files cannot be declared directly in an RPG source program; however, the FILE control statement can be used following compilation to change a random file to delayed-random. Other changes to the defaults generated by the RPG compiler are not recommended, and may cause undesired results during execution.

FILE Attribute Specifications

The FILE control statement allows modifications to be made to any of the disk file access technique attributes, whether or not these attributes can be specified in a source program. The syntax is as follows:

```
FILE <file-identifier> <attributes> ...;
```

The allowable <attributes> for use in specifying the disk file access technique can be selected from the following list:

<u>File Attribute</u>	<u>Abbreviation</u>
SERIAL	SER
RANDOM	RAN
DELAYED.RANDOM	D.R
EXTEND	EXT

When using an implied open (not allowed in the COBOL language), the attributes can be further defined by specification of the open type, as follows:

<u>File Attribute</u>	<u>Abbreviation</u>
INPUT	INP
OUTPUT	OUT
NEW	NEW

Any combination of these attributes can be used, where required, to specify the access technique desired. The keyword NO can be used with the EXTEND attribute or any of the implied open attributes to reset the specified attribute.

The NEW attribute is used to specify a new file to be created; if NEW is not specified, an old file (one that already exists in the disk directory) is expected. For example, specifying INPUT, OUTPUT, or INPUT OUTPUT (without NEW) causes an existing disk file to be opened and processed. Including the NEW attribute with the OUTPUT or INPUT OUTPUT attribute causes a new file to be created (NEW is ignored if OUTPUT is not specified).

Example:

```
?MODIFY TEST/PROGRAM  
?FILE FILE1 SERIAL INPUT NO OUTPUT;  
?FILE FILE2 RANDOM INPUT OUTPUT;  
?FILE FILE3 SERIAL INPUT OUTPUT EXTEND;  
?FILE FILE4 DELAYED.RANDOM;  
?FILE FILE5 NO EXTEND;
```


APPENDIX D FILE SECURITY

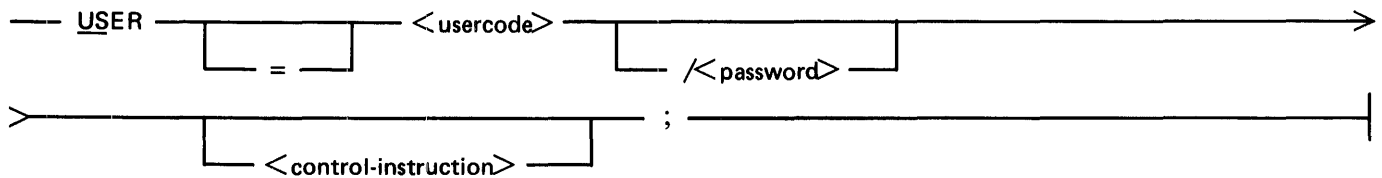
The file security system implemented on the B 1000 computer system is described in this section.

USER COMMAND

The USER command invokes the MCP file security system mechanism in the MCP and its associated naming convention.

The USER command causes the MCP to verify the usercode and password against the directory of valid usercodes and associated passwords in the (SYSTEM)/USERCODE file on the SYSTEM disk. The validated usercode is carried with <control-instruction> and is used to apply the MCP file security naming convention to any subsequent file-identifier references.

Syntax:



Semantics:

usercode

This field can be any valid usercode contained in the (SYSTEM)/USERCODE file.

password

This field can be any valid password that belongs to <usercode> and is contained in the (SYSTEM)/USERCODE file.

control-instruction

This field can be any valid MCP command.

Examples:

Command	Response
USER SITE/PRIV PD =/=;	PD = MASTER/(SITE)/FILE1 PD = MASTER/(SITE)/FILE2 PD = MASTER/(SITE)/FILE3 END PD.
PD (SITE)/=;	PD = MASTER/(SITE)/FILE1 PD = MASTER/(SITE)/FILE2 PD = MASTER/(SITE)/FILE3 END PD.
US STUDENT/JK EX TESTPROG PR 3; PD (FINANCE)/CHECKREG;	(STUDENT)/TESTPROG = 5 BOJ ... PD = (FINANCE)/CHECKREG END PD.
RE (FINANCE)/CHECKREG;	(FINANCE)/CHECKREG NOT REMOVED- SECURITY ERROR END REMOVE.
US FINANCE/VP REMOVE CHECKREG;	CHECKREG REMOVED END REMOVE.

FILE SECURITY SYSTEM

The file security system in the MCP is a mechanism for securing files secured against accidental or deliberate misuse. For example, a secured file cannot be listed, removed, or changed by an unauthorized user. A secured file is identified by the existence of the left and right parentheses "()" characters enclosing the first name.

Example:

(PAYROLL)/ <file-name >

A first name enclosed in the left and right parentheses "()" characters, such as (PAYROLL), is called the usercode. Rules exist for creating and accessing files having usercodes. Before files having usercodes can be created, a set of valid usercodes must be defined for the MCP. This is done with the SYSTEM/MAKEUSER program. The SYSTEM/MAKEUSER program creates a file on the SYSTEM disk called (SYSTEM)/USERCODE. The (SYSTEM)/USERCODE file contains all the valid usercode/password combinations that are allowed access to secured files.

The usercode allows a convenient way of naming and grouping sets of disk files. The password authorizes entry into the system.

File security is independent of terminal users and data communications. Thus, the usercode/password combination can be specified through the card reader, ODT, or remote terminals. The following discussion on file security is only in the context of batch processing.

File Security - Batch Processing

By executing the SYSTEM/MAKEUSER program, the (SYSTEM)/USERCODE file can be created or modified. Refer to the B 1000 Systems System Software Operation Guide, Volume 2, for a complete description of the operation of the SYSTEM/MAKEUSER program. The (SYSTEM)/USERCODE file contains information on all the valid usercode/password combinations, including the default packs, priority, and security information. This file is only accessed by the SYSTEM/MAKEUSER program and the MCP. Consequently, all access to the (SYSTEM)/USERCODE file can be prevented if the SYSTEM/MAKEUSER program is not stored on the SYSTEM or user disk.

After the (SYSTEM)/USERCODE file is created, the next step in creating secured files is the execution of a program with USER <usercode>/<password> prefixing the EXECUTE command.

Example:

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITER
```

In this example, the MCP executes the CHECK/WRITER program and uses PAYROLL/ACCT as the usercode/password combination.

OPERATION OF FILE SECURITY SYSTEM

The following paragraphs describe the operation of the file security system on the B 1000 computer system. In the examples, assume that the CHECK/WRITER program is executed with the USER PAYROLL/ACCT prefixing the EXECUTE command.

Control commands that are zipped by programs or entered through the card reader must be prefixed by a USER command with the appropriate usercode/password combination if the commands access secured files. Refer to the respective B 1000 language manuals for the syntax of the ZIP command.

Example 1:

The CHECK/WRITER program opens a new disk file labeled CHECKS, writes records into the file, and then closes it with LOCK (RELEASE in COBOL, SAVE IN COBOL74) file attribute. The file is entered in the disk directory. The MCP places into the disk directory the (PAYROLL)/CHECKS file name. The (PAYROLL)/CHECKS file is secured and can only be accessed by programs that are executed with the USER PAYROLL/<password> prefixing the EXECUTE command. When performing library maintenance commands (ADD, COPY, CHANGE, or REMOVE on this file, the command must be preceded by USER PAYROLL/<password>. The following are valid library maintenance commands for the (PAYROLL)/CHECKS files.

```
USER PAYROLL/ACCT RE CHECKS;  
USER PAYROLL/ACCT CH CHECKS TO PAYCHECKS;  
USER PAYROLL/ACCT COPY = FROM DISK TO BACKUP (KIND=TAPE);
```

The first command removes the (PAYROLL)/CHECKS file. The second command changes the file name from (PAYROLL)/CHECKS to (PAYROLL)/PAYCHECKS. The third command copies all the files with the (PAYROLL) first name, that is, (PAYROLL)/=, to a library tape labeled BACKUP. The COPY command causes the SYSTEM/COPY program to be executed.

The following commands can cause unexpected results.

```
RE CHECKS;  
RE (PAYROLL)/CHECKS;  
PD CHECKS;
```

In the first command, the MCP cannot find the (PAYROLL)/CHECKS file and, therefore, does not remove the file. If the CHECKS file exists on disk, it is removed. The second command is rejected by the MCP because the command requested the removal of a secured file. In the third command, the MCP cannot find the (PAYROLL)/CHECKS file, and if the CHECKS file exists on disk, it is displayed as being in the disk directory.

Example 2:

If another program is executed without the PAYROLL usercode (either a different usercode or no usercode) and it attempts to access the (PAYROLL)/CHECKS file, the MCP disallows the request.

Example 3:

If the user wishes the secured file to be accessed by other usercode/password combinations, the file must be designated as a PUBLIC file. In the previous examples, the (PAYROLL)/CHECKS file is created as a PRIVATE file by default and no unauthorized user can access it. If the (PAYROLL)/CHECKS file is a PUBLIC file, then any program executed with a different usercode/password combination or no usercode can access this file by referencing the full file name (PAYROLL)/CHECKS.

When a program attempts to open this file with the INPUT or INPUT/OUTPUT file attributes, the MCP checks to see if the file is a PUBLIC file, and if it is, opens it. The program can read from the file, write (if opened with INPUT/OUTPUT) to the file, and then close the file. The (PAYROLL)/CHECKS file then contains all updates made to it by the program.

Example 4:

The following are three ways of designating the security type, PUBLIC or PRIVATE, for a file.

1. Specify SECURITYTYPE = PUBLIC, SECURITYTYPE = PRIVATE, and SECURITYTYPE = DEFAULT in the FILE program attribute at execution time. The following example shows the use of the SECURITYTYPE file attribute.

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITER;  
FILE CHECKS SECURITYTYPE = <security>;
```

<security> can be the DEFAULT, PUBLIC, or PRIVATE keywords. The DEFAULT keyword specifies that the security type defined in the (SYSTEM)/USERCODE file is to be used.

2. After a file is created and locked in the disk directory, its security type can be changed by modifying the disk file header using the MH (Modify Header) system command. The following example shows the use of the MH system command to change the security type of the (PAYROLL)/CHECKS file.

```
USER PAYROLL/ACCT MH CHECKS SEC <security>;
```

<security> can be the PUBLIC or PRIVATE keywords.

3. The SECURITY system command can also be used to change the file security attributes. The following example shows the use of the SECURITY system command to change the security type of the (PAYROLL)/CHECKS file.

```
USER PAYROLL/ACCT SECURITY CHECKS <security>;
```

<security> can be the PUBLIC or PRIVATE keywords.

4. The SYSTEM/MAKEUSER program has an option that causes all files created by a specific usercode to be PUBLIC files. When creating the (SYSTEM)/USERCODE file, the PUBLIC keyword can be specified for any usercode/password combination. If the usercode/password combination is PUBLIC, then the MCP makes every file created by the usercode/password combination a PUBLIC file. If the PUBLIC keyword is not specified when the usercode/password combination is created, the default security type is PRIVATE.

The only programming languages available on the B 1000 computer system that allow files to be designated as PUBLIC or PRIVATE, are COBOL74 and COBOL74B. File declarations by other compilers specify DEFAULT for the security type for the usercode/password combination in the (SYSTEM)/USERCODE file. This default can be changed by the FILE attribute statement.

Example 5:

If a file is designated as a PUBLIC file, the creator of that file has the option of controlling the type of input/output (I/O) access that another user can perform (INPUT, OUTPUT, INPUT/OUTPUT) on that file. If the (PAYROLL)/CHECKS file is a PUBLIC read-only (INPUT) file, a program running under another usercode/password combination or no usercode can read the file but cannot write to it. Three ways of specifying the allowable I/O access are described next.

1. Specify the SECURITYUSE file attribute in the FILE program attribute in the EXECUTE command. The following example shows the use of the SECURITYUSE file attribute.

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITE;  
FILE CHECKS SECURITYUSE <access>;
```

<access> can be the INPUT, OUTPUT, or I.O keywords, and specifies read-only, write-only, or read-write, respectively.

2. The MH (Modify Header) system command can be used to change the I/O access of a file. The following example shows the use of the MH system command to change the I/O access of the (PAYROLL)/CHECKS file.

```
USER PAYROLL/ACCT MH CHECKS SUS <access>;
```

<access> can be the INPUT, OUTPUT, or I.O keywords and specifies read-only, write-only, and read-write, respectively.

3. The SECURITY system command changes the I/O access of a file. The following example shows the use of the SECURITY system command to change the I/O access of the (PAYROLL)/CHECKS file.

```
USER PAYROLL/ACCT SECURITY CHECKS PUBLIC <access>;  
USER PAYROLL/ACCT SECURITY CHECKS PRIVATE <access>;
```

<access> can be the IN, OUT, or IO keywords and specifies read-only, write-only, and read-write, respectively.

Example 6:

If the file security mechanism on the B 1000 computer system is not desired, all programs can be run without USER <usercode/password> prefixing the MCP commands with no noticeable impact. The (SYSTEM)/USERCODE file is not needed and is never accessed by the MCP. The following two restrictions apply when the file security mechanism is not used.

1. The first name portion of a file name cannot be enclosed within the parentheses "()" characters. A first name enclosed in the parentheses characters is assumed to be a usercode by the MCP.
2. The first name cannot begin with an asterisk (*) character. The use of the asterisk convention with file security is described in examples 7 and 8.

Example 7:

If a program is executed with the USER <usercode/password> prefixing the EXECUTE command and the program accesses a non-secured file that does not have a usercode as the first name, the following two conditions apply:

1. Existing files on disk having two names can be accessed by a program running under a usercode by specifying the exact title of the file. If a program running under a usercode attempts to close a new file with the LOCK file attribute set and the file has two names, the MCP does not enter the file in the disk directory. The file is not entered in the disk directory because the first name portion of the file name is used by the MCP for storing the usercode. Programs that create two-name files must be modified to use single file names if the program is executed with the USER <usercode>/<password> prefixing the EXECUTE command.

The CHECK/WRITER program can open the two-name NEW/INFO file as an output file, but this file cannot be closed and entered into the disk directory. No naming conflicts exist if the file is never locked in the disk directory. If the CHECK/WRITER program closes the file and desires to reopen it to access the records previously written, the CHECK/WRITER program must close the file without specifying any CLOSE options. The CHECK/WRITER program goes to end of job and the NEW/INFO file was not closed by the program, the NEW/INFO file is closed by the MCP and is not entered in the disk directory.

2. If the CHECK/WRITER program attempts to access an existing disk file labeled PAYABLES, the MCP first finds the file labeled (PAYROLL)/PAYABLES in the disk directory. If the (PAYROLL)/PAYABLES file is not in the disk directory, the MCP next searches for a file labeled PAYABLES. If the CHECK/WRITER program needs to access the PAYABLES file, it is possible only if there is no file in the disk directory labeled (PAYROLL)/PAYABLES.

If the CHECK/WRITER program declares the file name *PAYABLES, the MCP does not modify the file name in any way and searches for the single-name file labeled PAYABLES. The CHECK/WRITER program can read and/or write this file and, when it is closed, the file name remains unchanged.

A program executed with USER <usercode>/<password> prefixing the EXECUTE command cannot create a file with a single file name. The MCP prefixes such output file names with the usercode.

Example 8:

The SYSTEM/MAKEUSER program associates a default pack identifier with any usercode/password combination, subject to the condition that all like usercodes in the (SYSTEM)/USERCODE file must have the same pack identifier and security type (PUBLIC or PRIVATE), but different passwords.

If the pack identifier specified in the program file declaration is blank, the file is automatically directed to the user disk specified by the usercode entry in the (SYSTEM)/USERCODE file. If a pack identifier is included in the file declaration, it overrides the default pack identifier specified in the (SYSTEM)/USERCODE file. The following example illustrates this.

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITER;  
FILE CHECKS PACK.ID PAYR;
```

In this example, the MCP directs the file labeled CHECKS to the user disk labeled PAYR, creating a file labeled PAYR/(PAYROLL)/CHECKS, irrespective of the default pack identifier contained in the (SYSTEM)/USERCODE file. If a default pack identifier of BACKUP is contained in the (SYSTEM)/USERCODE file and the CHECK/WRITER program is executed without the FILE program attribute specified in the EXECUTE command (the user disk name in the CHECKS file is blank), the MCP directs the file to the default user disk labeled BACKUP with the resulting name of BACKUP/(PAYROLL)/CHECKS.

If a default pack identifier is specified in the (SYSTEM)/USERCODE file and it is necessary to create the file on the SYSTEM disk, then a pack name of DISK must be supplied.

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITER;  
FILE CHECKS NAME DISK/(PAYROLL)/CHECKS;
```

Whenever the MCP detects an asterisk (*) character in the first position of the declared first name, the MCP deletes the asterisk (*) character and uses the resulting file name without further modification.

Example 9:

There are programs, such as Message Control Systems (MCS) and utility programs, that need to access and create files with any usercode. To permit this, a usercode/password combination can be defined in the (SYSTEM)/USERCODE file as a privileged (*PRIV) usercode.

The following declared and actual file names assume that (PAYROLL) is a privileged usercode with a default pack identifier of PAYR, and the usercode (LEDGER) is a non-privileged usercode with a default pack identifier of LEDG. If the CHECK/WRITER program attempts to create a new file, the MCP modifies the file name as in table D-1.

Table D-1. File Names Created by the CHECK/WRITER Program.

Declared File Name	Actual File Name Used by the MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
*CHECKS	PAYR/CHECKS/
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK
*CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK
(LEDGER)/CHECKS	LEDG/(LEDGER)/CHECKS
DISK/(LEDGER)/CHECKS	DISK/(LEDGER)/CHECKS

PAYR/CHECKS/PAYCHECK PAYR/CHECKS/PAYCHECK

If the CHECK/WRITER program attempts to access an existing file, the MCP uses the actual file name as shown the table D-2.

Table D-2. Actual File Names Used by the MCP

Declared File Name	Actual File Name Used by the MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK

All other cases of accessing existing files function the same as shown in table D-1 for creating new files.

Non-Privileged Usercode Handling

Non-privileged usercodes are restricted in the file names that can be accessed. The following examples assume that the usercode (PAYROLL) is a non-privileged usercode with a default pack identifier of PAYR.

Example 1:

If the CHECK/WRITER program attempts to create a new file, the MCP modifies the file name as shown in table D-3.

Table D-3. Actual File Names Used When Creating a New File

Declared File Name	Actual File Name Used by the MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
DISK/(PAYROLL)/CHECKS	DISK/(PAYROLL)/CHECKS

If the CHECK/WRITER program attempts to access an existing file, The MCP modifies the file name as shown in table D-4.

Table D-4. File Names Used When Accessing an Existing File

Declared File Name	Actual File Name Used by the MCP
CHECKS	PAYR/(PAYROLL)/CHECKS
*CHECKS	PAYR/CHECKS/ (see NOTE)
CHECKS/PAYCHECK	PAYR/CHECKS/PAYCHECK (see NOTE)
DISK/CHECKS/PAYCHECK (PAYROLL)/CHECKS	DISK/CHECKS/PAYCHECK (see NOTE) PAYR/(PAYROLL)/CHECKS
DISK/(PAYROLL)/CHECKS	DISK/(PAYROLL)/CHECKS
BACKUP/(PAYROLL)/CHECKS	BACKUP/(PAYROLL)/CHECKS
(LEDGER)/CHECKS	LEDG/(LEDGER)/CHECKS (see NOTE)
DISK/(LEDGER)/CHECKS	DISK/(LEDGER)/CHECKS (see NOTE)

NOTE

Access is granted only if the file is a PUBLIC file and the SECURITYUSE
<access> matches the open type.

Example 2:

If a program is executed with the USER <usercode>/<password> prefixing the EXECUTE command and then zips a control command to the MCP, the MCP automatically prefixes the control command with USER <usercode>/<password> of the zipping program. For example, ZIP "RE (LEDGER)/CHECKS;" is interpreted by the MCP as USER PAYROLL/ACCT RE (LEDGER)/CHECKS and is not allowed unless the usercode (PAYROLL) is a privileged usercode.

If the zipping program includes USER <usercode>/<password> in the control command, the usercode/password combination overrides the combination added by the MCP. For example, ZIP USER LEDGER/FILE REMOVE CHECKS; removes the file, even if the usercode (PAYROLL) is non-privileged. This allows one non-privileged program to remove the files of another, but only if the correct usercode/password combination is known by the zipping program.

Example 3:

The usercode/password information must be prefixed to an ODT system command if the command results in the modification or removal of a secured disk file. The following system commands are affected by this rule.

CHANGE
COMPILE
EXECUTE
MH
MODIFY
QF
REMOVE

The following examples are of incorrect and correct system commands.

Incorrect Messages

EXECUTE (LEDGER)/XXX;
CHANGE (LEDGER)/A TO X;
USER LEDGER/FILE CHANGE A TO *X;

Correct Messages

USER LEDGER/FILE EXECUTE (LEDGER)/X;
or
USER LEDGER/FILE EXECUTE X;
USER LEDGER/FILE CHANGE A TO X;
USER SITE/X CHANGE (LEDGER)/A TO *X;
(if (SITE) is a privileged usercode)

Example 4:

Backup files created by programs executed with USER <usercode>/<password> prefixing the EXECUTE command have a naming convention that is different from the non-secured backup file naming convention. For example, backup files created by the CHECK/WRITER program have the following names, assuming a default backup pack identifier of PAYR.

Printer Backup	Punch Backup
PAYR/(PAYROLL)/PRT<integer>	PAYR/(PAYROLL)/PCH<integer>

To print, punch, remove, or display secured backup files, the PB, RB, and BF commands must be preceded by the appropriate USER <usercode>/<password>. The following examples include the system command and the action taken for secured backup files.

System Command	Action
USER PAYROLL/ACCT PB 15;	Prints PAYR/(PAYROLL)/PRT15 or Punches PAYR/(PAYROLL)/PCH15.
USER PAYROLL/ACCT BF PRT/=;	Displays all non-user named printer backup files on user disk PAYR with usercode (PAYROLL).
USER PAYROLL/ACCT RB =/=;	Removes all non-user named backup and dump files on user disk PAYR with usercode (PAYROLL).

Example 5:

The usercode/password convention is applicable to the ADD and COPY commands. For example, the command

```
USER PAYROLL/ACCT COPY = FROM PAYR(KIND=DISK) TO SYSTEM (KIND=TAPE);
```

copies (PAYROLL)/= from the user disk labeled PAYR to the library tape labeled SYSTEM.

Example 6:

Programs executed with USER <usercode>/<password> prefixing the EXECUTE command are also subject to the default pack identifier specified in the (SYSTEM)/USERCODE file. Assume that the default pack identifier for the usercode (PAYROLL) is PAYR, and then consider the following example.

```
USER PAYROLL/ACCT EXECUTE CHECK/WRITER;
```

The MCP first searches for the program file PAYR/CHECK/WRITER and, if found, executes it. If the program cannot be found on the default user disk, the SYSTEM disk is searched. If program files of the same name exist on both the default user disk and the SYSTEM disk, the file on the SYSTEM disk is never loaded. To overcome this restriction, a pack of DISK can be used as shown in the following example:

```
USER PAYROLL/ACCT EXECUTE DISK/CHECK/WRITER;
```

The same restrictions occur when the program being executed is identified by a single file name as shown in the following example.

```
USER PAYROLL/ACCT EXECUTE PAYCHECKS;
```

In this case, the following search sequence is followed:

```
PAYR/(PAYROLL)/PAYCHECKS  
DISK/(PAYROLL)/PAYCHECKS  
PAYR/PAYCHECKS  
DISK/PAYCHECKS
```


APPENDIX E

NOTATION CONVENTIONS

The following paragraphs describe the notation conventions used in this manual.

LEFT AND RIGHT BRACKETS ([])

Left and right bracket characters enclose portions of a particular syntax that is not required.

LEFT AND RIGHT BROKEN BRACKETS (< >)

Left and right broken bracket characters enclose letters and digits that are supplied by the user. The letters and digits can represent a variable, a number, a file name, or a command.

Example:

<mix #>AX<command>

AT SIGN (@)

The at sign (@) character encloses hexadecimal information.

Example:

@F3@ is the hexadecimal representation of the EBCDIC character 3.

The at sign (@) character also encloses binary information when the initial @ character is followed by a (1).

Example:

@(1)11110011@ is the binary representation of the EBCDIC character 3.

RAILROAD DIAGRAMS

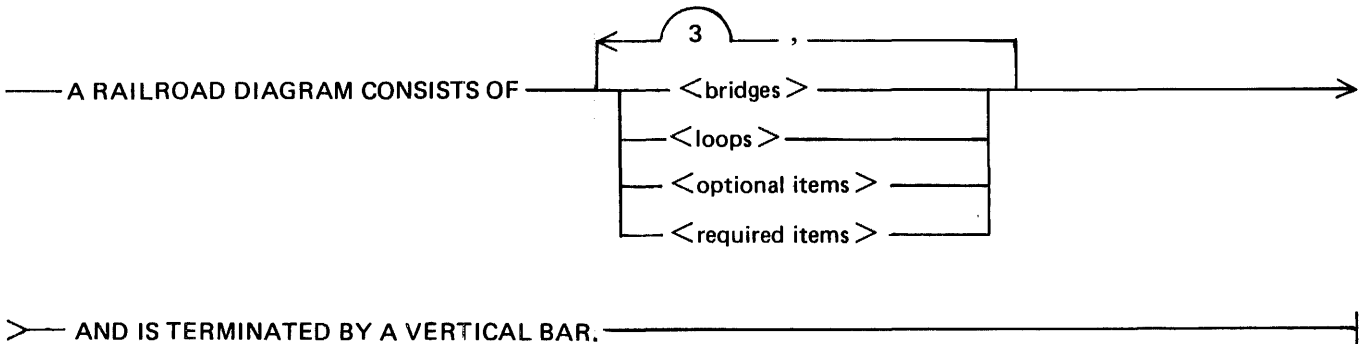
Railroad diagrams show how syntactically valid statements can be constructed.

Traversing a railroad diagram from left to right, or in the direction of the arrowheads, and adhering to the limits illustrated by bridges produces a syntactically valid statement. Continuation from one line of a diagram to another is represented by a right arrow (→) appearing at the end of the current line and beginning of the next line. The complete syntax diagram is terminated by a vertical bar (|).

Items contained in broken brackets (< >) are syntactic variables which are further defined, or require the user to supply the requested information.

Upper-case items must appear literally. Minimum abbreviations are underlined

Example:



G50051

The following syntactically valid statements can be constructed from the above diagram:

A RAILROAD DIAGRAM CONSISTS OF <bridges> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <optional-items> AND IS TERMINATED BY A VERTICAL BAR.

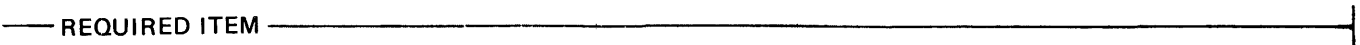
A RAILROAD DIAGRAM CONSISTS OF <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

A RAILROAD DIAGRAM CONSISTS OF <optional-items>, <required-items>, <bridges>, <loops> AND IS TERMINATED BY A VERTICAL BAR.

Required Items

No alternate path through the railroad diagram exists for required items or required punctuation.

Example:

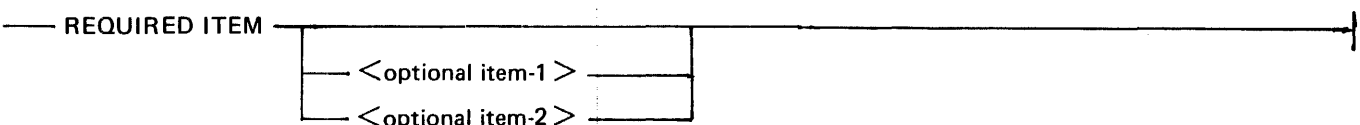


G50052

Optional Items

Items shown as a vertical list indicate that the user must make a choice of the items specified. An empty path through the list allows the optional item to be absent.

Example:



G50053

The following valid statements can be constructed from the above diagram:

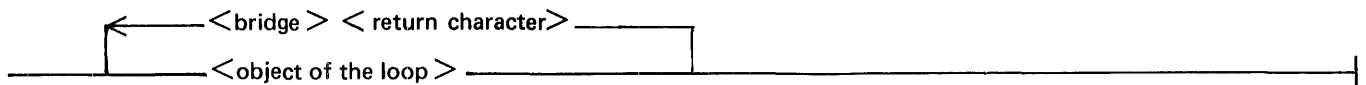
REQUIRED ITEM

REQUIRED ITEM < optional-item-1 >

REQUIRED ITEM < optional-item-2 >

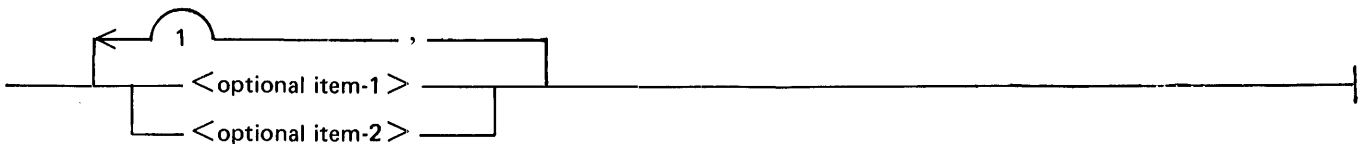
Loops

A loop is a recurrent path through a railroad diagram. A loop must be traversed in the direction of the arrowheads, and the limits specified by the bridges cannot be exceeded. A loop has the following format:



G50054

Example:



G50055

The following statements can be constructed from the previous railroad diagram:

< optional-item-1 >

< optional-item-2 >

< optional-item-1 > , < optional-item-1 >

< optional-item-1 > , < optional-item-2 >

< optional-item-2 > , < optional-item-1 >

< optional-item-2 > , < optional-item-2 >

A loop must be traversed in the direction of the arrow heads, and the limits specified by bridges cannot be exceeded.

Bridges

A bridge indicates the minimum or maximum number of times a path can be traversed in a railroad diagram. There are two forms of bridges.



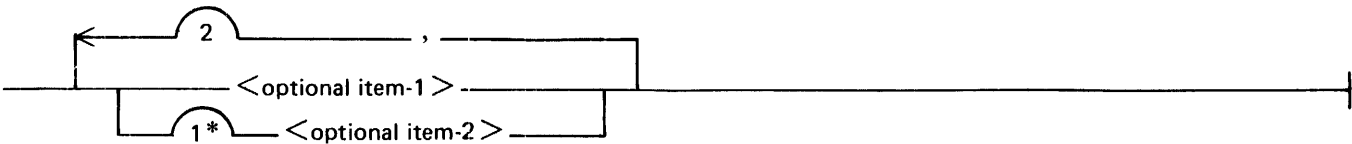
n is an integer which specifies the maximum number of times the path can be traversed.



G50056

n is an integer which specifies the minimum number of times the path must be traversed.

Example:



G50057

The loop can be traversed a maximum of two times; however, the path for <optional-item-2> must be traversed at least one time.

The following statements can be constructed from the previous railroad diagram:

<optional-item-2>

<optional-item-1> , <optional-item-2>

<optional-item-2> , <optional-item-2> , <optional-item-1>

<optional-item-2> , <optional-item-2> , <optional-item-2>

Words appearing in all upper case are keywords and must appear as presented. Words in lower case and embraced in left and right broken bracket characters are symbols for user-supplied words or values. Where keywords can be abbreviated, the abbreviation is denoted by the letters underlined in the keywords. Each keyword, keyword abbreviation, or user-supplied item must be delimited from another such word by at least a single space or by a special character. For example, the parentheses, comma, period, colon characters can be used as delimiters. Redundant spaces are ignored.

INDEX

<abort-reference> 7-2
 <date> 7-3
 <family-name> 1-7
 <file-title> 5-144
 <firmware-file-title> 6-6
 <first-name> 1-7
 <line-address> 6-3
 <line-id> 6-3
 <line-name> 6-3
 <line-number> 6-3
 <mm>/<dd>/<yy> 5-35, 5-37
 <priority> 7-2
 <second-name> 1-7
 <start parameter list> 5-144
 <station-id> 6-4
 <task equation list> 5-144
 <task-specifier> 7-1
 <time> 7-3
 <usercode> 1-7
 (1), (2), (3), (4) 5-24, 5-151
 + 5-3, 5-55, 5-73
 +, -, *, /, MOD 5-24
 * 1-7
 *DEFAULT 5-87
 *NULL 5-87
 ; 5-10
 - 5-3, 5-55, 5-73
 = 5-24, 5-43, 5-45, 5-49, 5-50, 5-58, 5-99
 =/= 5-14, 5-16, 5-58, 5-99, 5-101, 5-112,
 5-115, 5-119, 5-125, 5-162, 5-166
 " " 5-12, 5-148
 AB 5-2
 AB <integer> 5-2
 AB (Auto Backup) 2-2
 AB (AUTOBACKUP) 5-1
 AB (Automatic Backup) 2-9
 AB +<unit-mnemonic> 5-2
 AB -<unit-mnemonic> 5-2
 ABNORMALSAVE 4-88
 AC 4-1
 AC (Accept) 2-2
 AC (Response to Accept Message) 5-5
 ACCESSROUTINES or ACR 5-134
 ACTIVE 5-162
 Active Schedule 1-3
 ADD 5-6
 AFTER 4-2
 AFTER.NUMBER 4-3
 AL (Align Forms) 2-7, 2-9

INDEX (CONT)

AL (Align Printer) 5-7
ALL 5-40, 5-85, 5-167
ALLOCATE.AT.OPEN 4-20
AMCS 3-8
AP (Auto Print) 5-8
AP (Autoprint) 2-9
AREAS 4-21
ASCII 4-22, 5-137, 5-138
AT 5-9
At Sign (@) E-1
AUDIT 5-135
AUDITED 4-23
AUDITED and PROTECTED Files C-6
AUDITED FILES C-6
AUDIT0, AUDIT1, and AUDIT2 6-5
AUTOPRINT 4-24
AX 4-4
AX (Accept) 2-2
AX (Response to Accept Message) 5-10
B 1000 MEMORY MANAGEMENT ALGORITHMS A-4
B 1247 Train Printer Control (Control
Identification = @10@) 5-74
B 1247-4 Train Printer Control (Control
Identification = @3E@) 5-75
BACK 5-60
BACKUP 4-25, 5-32
BACKUP.DISK 4-26
BACKUP.TAPE 4-27
Bar Graph B-7
Bar Graph Scale Factor B-7
BASIC File Declarations C-8
BB (Backup Blocks Per Area) 2-9
BB (Backup Blocks per Area) 5-11
BB (Backup Blocks) 2-2
BCL 4-28
BD (Backup Designate) 5-12
BD (Backup Disk) 2-2, 2-9
BF (Backup Files) 2-2, 2-9
BF (Display Backup Files) 5-13
BINARY 4-29
bit-string 5-23, 5-151
BLOCKS.PER.AREA 4-30
BOJ 3-8
BREL 3-8
Bridges E-4
BUFFERS 4-31
Burroughs Network Architecture (BNA)
Commands 2-11
C 5-165

INDEX (CONT)

CANCEL 5-87
 CARD.PUNCH 4-32
 CARD.READER 4-33
 CASSETTE 4-34
 Cassettes, How to Create 3-1
 CC (Control Card) 5-15
 CD (Card Decks) 2-2, 2-10
 CD (List Card Decks in Pseudo Readers) 5-16
 Central Service Module 1-2
 CHANGE 5-17
 Changing the Status of Programs Running
 in the Mix 2-8
 Changing the System Software Environment 2-11
 character-string 5-23
 CHARGE 4-5
 CHR9 3-9
 CL (Clear Unit) 2-6, 2-7, 5-19
 clear/start Cassette, Creating the 3-1
 clear/start Operation, How to 3-6
 Clear/Start the New System Software 3-17
 clear/start With OLD/Software 3-15
 CLOS 3-9
 CLOSE Command 6-5
 CM (Change System Software) 2-11, 5-20
 COBOL File Declarations C-8
 CODE 4-49
 COLDSTART COMPLETE - CLEAR/START REQUIRED 3-5
 Coldstart Tape Halts 8-16
 COLDSTART/DISK Cassette, Using the 3-4
 COLDSTART/DISK MARK <mark-level>.<patch-level>
 <date + compile time> 3-5
 COLDSTART/TAPE Cassette, Using the 3-3
 column 5-97
 command 6-9
 Command entry 5-1
 Common Syntax 6-3
 COMPILE 4-6
 compile-or-execute 4-8, 4-168, 4-171
 compile-syntax 4-1 thru 4-5, 4-8, 4-13,
 4-18, 4-133 thru 4-138,
 4-140, 4-141, 4-143, 4-144,
 4-146, 4-147, 4-148, 4-150,
 4-152, 4-154, 4-155, 4-159,
 4-161, 4-164, 4-168, 4-169,
 4-171, 4-173, 4-174, 4-175
 compiler-name 1-8, 4-7, 5-97
 COMPRESS 5-61
 CONCEPTS AND DEFINITIONS A-1
 CONDITIONAL 4-8

INDEX (CONT)

Control Instructions 1-4
 Control of Pseudo Card Readers 2-10
 control-attribute 5-110, 5-111
 control-instruction 5-158, D-1
 Controlling Backup (Spooling) Functions 2-9
 Controlling Programs in the Schedule 2-8
 CONTROLPOINT 5-135
 COPIES 5-98
 COPY 3-9, 4-35, 5-21
 Copy New System Files 3-15
 Copy the Software to be Replaced 3-14
 CP (Compute) 2-8, 5-22
 CP/= 5-166
 CPY 5-166
 CQ (Clear Queue) 5-25
 CQ (Clear the Queue) 2-7
 CREATE 6-6
 Creating the clear/start Cassette 3-1
 CST and PST A-9
 CT (Chip Table) 5-26
 CU (Core Usage) 2-1, 2-2, 5-27
 CX 5-165
 DATA 4-9, 4-49
 Data Communications ODT 6-20
 DATA.RECORDER.80 4-36
 data-base-name 5-114, 5-136
 DATARATE Command 6-7
 DATE 3-9
 Date and Time 3-7
 DB (Data Base Status) 2-1
 DB (Data Base) 2-2, 5-29
 DEBUG 3-9
 DC 5-64, 5-66
 DECAY.INTERVAL A-9
 DEF 5-61
 DEFAULT 4-37, 4-107
 DELAY 5-61
 Delayed Random Files C-5
 DELAYED.RANDOM 4-38
 Determine Current Operating Environment 3-13
 Device Requirements 1-1
 DF (Date of File) 2-2, 2-5, 5-30
 Differences Between Data Communication and
 ODT-Control ODT Device 6-20
 DIR 5-31
 DIRECTION 5-61
 DISK 4-39, 5-141
 Disk File Access Characteristics C-2
 Disk File Access Specifications C-7

INDEX (CONT)

Disk File Header C-2
Disk Operation and Interrogate Messages 2-5
Disk Pack, How to Initialize 3-2
DISK.CARTRIDGE 4-40
DISK.FILE 4-41
DISK.PACK 4-42
disk-identifier 4-81
DISP 3-9
DK 5-64, 5-66
DKx 5-142
DL (Disk Location) 2-2, 2-9, 5-32
DM (Dump Memory and Continue) 2-6, 5-33
DMP/= 5-13, 5-112, 5-118
DOUBLE 5-98
DP 5-64, 5-66
DP (Dump and Discontinue) 2-6, 2-8
DP (Dump and DS) 2-4
DP (Dump Memory and Discontinue) 5-34
DP System Command 1-4
DR (Change MCP Date) 5-35
DR (Date Reset) 2-1
DRIVE 4-43
DS (Discontinue Program) 5-36
DS (Discontinue) 2-4, 2-8
DS System Command 1-4
DSKAVL 5-58
DT (Change MCP Date) 5-37
DUMMY.FILE 4-44
DUMP 3-9, 5-32
DUMP Command 6-8
DYNAMIC 4-10
DYNAMIC.SPACES 4-12
EBCDIC 4-45
ED (Eliminate Decks) 2-10
ED (Eliminate Pseudo Deck) 5-38
EM (ELOG Message) 5-39
EM (Engineers Message) 2-10
END 4-14
END.OF.PAGE 4-46
ENDCTL 4-16
ending-record 5-97
ENTER INPUT DRIVE - <DC?, DP? OR DKA> 3-5
ENTER OUTPUT DRIVE - <DCA, DPA OR DKA> 3-5
EOJ 3-9, 5-143
EQUAL 5-98
ER (Error Rate) 2-10, 5-40
ET (ELOG Transfer) 5-41
ET (Engineers Log Transfer) 2-10
EVEN.PARITY 4-47

INDEX (CONT)

EXECUTE 4-17
 Experimental Name Table Entries 3-17
 EXTEND 4-48
 EXTEND Attribute Reset C-4
 EXTEND Attribute Set C-3
 Extended Segment Decay A-7
 Extended Segment Decay Advantages A-8
 Extended Segment Decay Disadvantages A-8
 external-file-name 5-43
 family-name 4-136, 5-12, 5-14, 5-30, 5-32,
 5-58, 5-97, 5-101, 5-107, 5-112,
 5-114, 5-116, 5-118, 5-120, 5-136,
 5-141, 5-148
 FATALERROR-ENABLED or FATALERROR 5-135
 FILE 4-18
 FILE ALL 6-18
 FILE Attribute Specifications C-11
 File Attributes 4-19
 file identifier 5-97
 File Information Block C-1
 File Security - Batch Processing D-3
 File Security System D-2
 FILE.TYPE 4-49, 5-77
 file-attribute 4-19 thru 4-34, 4-36 thru 4-49,
 4-51 thru 4-76, 4-79 thru 4-132
 file-id-1 5-18
 file-id-2 5-18
 file-identifier 1-6, 4-9, 4-14, 4-163, 4-167,
 5-8, 5-20, 5-30, 5-56, 5-114,
 5-127
 file-name 1-7
 file-title 1-7
 FIRMWARE 6-5
 FIRMWARE HALTS (L=@00FOxx@ OR @000Fxx@) 8-2
 first-name 5-30, 5-58, 5-97, 5-101, 5-116
 first-name-1 5-18
 first-name-2 5-18
 Fixed Lamp Display B-9
 FLEXIBLE 4-51
 FLMP 3-10, B-2
 FM (Forms Mounted) 2-2
 FM (Response to Special Forms) 5-42
 FN (Display File Name) 5-43
 FN (File Name) 2-2, 2-5
 FOOTING 4-52
 FORMS 4-53
 FORTRAN File Declarations C-8
 FR (Final Reel of Unlabeled Tape File) 5-44
 FR (Final Reel) 2-6

INDEX (CONT)

FREEZE 4-133
FS (Force from Schedule) 2-8
FS (Force from Waiting Schedule) 5-45
FS System Command 1-3
FUNCTIONAL CHARACTERISTICS A-8
G 5-165
General Specifications 3-7
Generic Terms 1-6
GISMO HALTS (L=@0D00xx@) 8-3
GO 4-7
GO (Go) 2-8
GO (Resume Stopped Program) 5-46
GX 5-165
HALT 5-47
Handling Magnetic Tape Devices 2-6
Handling Memory Dumps 2-6
Handling Peripherals 2-7
Handling the Log Files 2-10
Handling the System Options 2-6
HARDWARE 4-54
Hardware Initialization 3-1
HEADER 4-55
HELP 5-61
HELP Command 6-9
hex-number 5-146
hex-string 5-23, 5-151
hexadecimal-number 4-165
HN (Host Name) 2-3, 2-11
HN (Hostname) 5-48
HOLD 4-134
host-name 5-56
HOSTNAME 4-56
hostname 5-9, 5-48, 5-97
HOSTNAME 5-98
How to clear/start the Operating System 3-6
How to Coldstart the Operating System 3-3
How to Create the System Cassettes 3-1
How to Initialize a Disk Pack 3-2
How to Initialize the Operating System 3-3
How to Set the Operating System Options 3-7
HS (Hold in Schedule) 2-9
HS (Hold in Waiting Schedule) 5-49
HS System Command 1-3
HW (Hold in Waiting Schedule until Job EOJ) 5-50
HW (Hold in Waiting Schedule) 2-9
HW System Command 1-3
I 5-165
I.O 4-108
I/O Descriptors and File Buffers C-1

INDEX (CONT)

IB (Instruction Block) 5-51
 IC (Interpreter Count) 2-3, 2-6, 5-52
 ID 5-61
 identifier 4-166
 IL (Ignore Label) 2-4, 2-5, 2-7, 2-10, 5-53
 IMPLIED.OPEN.DENIAL.FLAG 4-57
 IN 5-132
 INDEX.SEQUENTIAL 4-49
 Initializing the Operating System 3-3
 INP 5-61
 INPUT 4-58, 4-108
 Input Serial Files C-2
 Input/Output Serial Files C-3
 INSTALLING NEW SYSTEM SOFTWARE UPDATES 3-13
 integer 4-5, 4-13, 4-21, 4-30, 4-31, 4-43,
 4-52, 4-60, 4-61, 4-64, 4-65, 4-66,
 4-67, 4-68, 4-74, 4-82, 4-89, 4-91,
 4-92, 4-95, 4-96, 4-99, 4-101, 4-103,
 4-106, 4-125, 4-141, 4-143, 4-144,
 4-150, 4-153, 4-156, 4-159, 4-162,
 4-165, 4-169, 4-175, 5-3, 5-8, 5-11,
 5-13, 5-16, 5-23, 5-26, 5-38, 5-52,
 5-53, 5-79, 5-82, 5-105, 5-108, 5-109,
 5-112, 5-115, 5-118, 5-122, 5-129,
 5-131, 5-134, 5-140, 5-142, 5-146,
 5-151, 5-155, 5-157
 integer-1 5-64, 5-66, 5-80, 5-133, 5-168,
 5-169
 integer-2 5-64, 5-66, 5-81, 5-133, 5-168,
 5-169
 integer-3 5-64, 5-66, 5-81, 5-168, 5-169
 internal-file-identifier 4-19 thru 4-34,
 4-36 thru 4-49,
 4-51 thru 4-76,
 4-79 thru 4-132
 internal-file-identifier1 4-35
 internal-file-identifier2 4-35
 INTERPRETER 4-49, 4-135
 interpreter-name 1-8
 Interpreters 1-2
 Interrogating and Changing the Time and Date 2-1
 Interrogating the Status of Programs in the
 Mix 2-1
 Intializing a Disk Pack, How to 3-2
 INTNAME 4-59
 INTRIN.DIRECTORY 4-136
 INTRINSIC 4-49
 INTRINSIC.NAME 4-137
 intrinsic-identifier 4-137

INDEX (CONT)

INVALID.CHARACTERS 4-60
INVISIBLE 4-138
Invoking The Performance Monitoring
 Capability B-2
IO 5-132
IOLOG Command 6-10
IS COMPLETE COPY DESIRED? <YES OR NO> 3-5
IS DATA COMPARISON DESIRED? <YES OR NO> 3-5
IV (Invisible) 5-55
IX 5-165
JOBS 5-27
JOBSTART 5-56
JOBSTART (Initiate Job Transfer to Another
 Host) 2-11
JS (Jiggle Schedule) 2-9, 5-57
JS System Command 1-3
KA (Analyze Disk Directory) 5-58
KA (Analyze Disk) 2-5
KB (Keyboard Options) 2-6, 2-7
KB (Print ODTLOG) 5-60
KC (Print Disk Segments in Character
 Format) 2-5, 5-64
KEY 5-98
KEYCOMPARE 5-135
KP (Print Disk Segments in Hexadecimal
 Format) 5-66
KP (Print Packed Disk Segment) 2-5
LAB 3-10
LABEL.TYPE 4-61
LABELS 5-98
LC (Log Comment) 2-10, 5-68
LD (Load Control) 2-10
LD (Pseudo Load) 5-69
Left and Right Brackets ([]) E-1
Left and Right Broken Brackets (<>) E-1
length 5-97
LEVEL 4-139
Level One (First-In, First-Out) A-4
Level One Advantages A-4
Level One Disadvantages A-4
Level Three (Memory Priority With Thrashing
 Detection) A-6
Level Three Advantages A-7
Level Three Disadvantages A-7
Level Two (First-In, First-Out With Thrashing
 Detection) A-5
Level Two Advantages A-6
Level Two Disadvantages A-6
LG (Transfer and Print Log) 5-71

INDEX (CONT)

LIB 3-10
 LIBRARY 4-7
 LINE ALL 6-18
 LINEFORMAT 4-62
 LN (Transfer and Print Log) 5-72
 LN or LG (Logout) 2-10
 Loading a New SDL2 Interpreter 3-14
 Loading a New SDL2 Intrinsic File 3-15
 Loading SDL2 Intrinsic File 3-16
 Loading SYSTEM/COPY 3-16
 LOCK 4-63
 LOG 3-10
 log-message 5-68
 Logical Versus Physical I/O C-1
 logical-station-number 4-114
 Loops E-3
 LOWER.MARGIN 4-64
 LP 5-61
 LP (Lock Program) 2-8
 LP (Lock Protection) 5-73
 lsn 6-4
 LT (Load Train Printer Table) 2-7
 LT (Load Translator) 5-74
 M 5-165
 MAKE Command 6-11
 MAX.SWEEP.INTERVAL A-8
 MAXIMUM.BLOCK.SIZE 4-65
 MAXRECSIZE 4-66
 MAXSUBFILES 4-67
 MAXWAIT 4-140, 5-135
 MC/= 5-166
 MCP HALTS (L=@000011@) 8-6
 MCP Options 3-8
 MCP Performance Monitoring Options B-2
 MCP-option 5-123, 5-139, 5-154
 MCS 5-165
 MCX 5-166
 MEM 3-10
 MEM.EXTEND.CLOCK A-9
 MEM.EXTEND.COUNT A-9
 MEMORY 4-141
 Memory Fragmentation A-1
 MEMORY.PRIORITY 4-142, A-9
 MEMORY.STATIC 4-144
 message 4-1, 4-4, 5-5, 5-9, 5-10, 5-39, 5-89
 Messages Concerning Program Status 2-4
 MH (Modify Header) 2-5, 5-77
 MICRO MCP HALTS (L=@0200xx@) 8-5
 MINRECSIZE 4-68

INDEX (CONT)

mix number 1-9
 mix-number 4-3, 4-5, 4-8, 4-10, 4-13, 4-19,
 4-35, 4-133, 4-135, 4-136, 4-137,
 4-139, 4-140, 4-141, 4-143, 4-144,
 4-146, 4-147, 4-149, 4-150, 4-152,
 4-155, 4-159, 4-161, 4-164, 4-169,
 4-171, 4-173, 4-174, 4-175, 5-5,
 5-10, 5-27, 5-29, 5-33, 5-34,
 5-36, 5-42, 5-43, 5-44, 5-45,
 5-46, 5-49, 5-53, 5-55, 5-73,
 5-82, 5-83, 5-85, 5-90, 5-91,
 5-94, 5-103, 5-108, 5-109, 5-111,
 5-121, 5-129, 5-140, 5-145, 5-146,
 5-152, 5-156, 5-157, 5-162, 5-167
 mix-number-1 5-50, 5-143
 mix-number-2 5-50, 5-143
 mix-number1 5-125
 mix-number2 5-125
 mix-number3 5-125
 ML (Mix Limit) 2-3, 2-8, 5-79
 ML System Command 1-4
 MM 5-80, 5-165
 MM (Memory Management) 2-8
 MMX 5-165
 MODE 5-62
 MODIFY 4-145
 MP (Memory Priority) 2-3, 2-8, 5-82
 MPRI 3-10
 MR (Close Output File with Purge) 5-83
 MR (Most Recent File) 2-4
 MU (List Multipack File Tables) 5-84
 MU (Multipack File) 2-3, 2-5
 MULTI.PACK 4-69
 MX 5-165
 MX (Mix and Status Interrogation) 5-85
 MX (Mix) 2-1, 2-3
 MX System Command 1-4
 MY.NAME 4-70
 N 5-164
 NAME 4-71
 NC (Network Controller Command) 2-8
 NC (Network Controller) 5-86
 NDL Handler 3-17
 NET (BNA Network Mode Inquiry) 2-3
 NET (Network Mode Change or Inquiry) 2-11, 5-87
 Network Controller 6-1
 Network Controller Execution 6-1
 Network Controller Input Commands 6-3
 Network Controller Priority 6-1

INDEX (CONT)

Network Controller Program Switches 6-1
NEW.FILE 4-72
new-address 6-15
new-file-name 4-59
NO.DEATH.IN.FAMILY 4-146
NOLIST 5-98
Non-Privileged Usercode Handling D-8
NOT 4-73, 6-11
NOW 5-87
NUMBER.OF.REMOTE.STATIONS 4-74
number-of-copies 5-97
NW (Network Message) 2-11, 5-89
NX 5-165
O/= 5-166
OBJ 4-7, 4-11, 4-147
octal-string 5-23, 5-151
ODD 4-75, 5-137, 5-138
ODT 5-71, 5-72, 5-153, 5-166
ODT Commands 6-5
ODTL 3-11
ODX 5-166
OF (Optional File Response) 5-90
OF (Optional File) 2-2
OFF 5-81, 6-10, 6-17
OK (Continue Processing) 5-91
OK (OK) 2-4
OL (Display Peripheral Status) 5-92
OL (Output Label) 2-3, 2-5, 2-7
ON 5-81, 5-135, 6-10, 6-17
OPEN 3-11
OPEN.LOCK 4-76
OPEN.LOCKOUT 4-77
OPEN.ON.BEHALF.OF 4-78
Operating System, How to clear/start 3-6
Operating System, How to Coldstart 3-3
Operating System, How to Initialize 3-3
Operating System, How to Set Options 3-7
Operation of File Security System D-3
Operational Details B-2
Operator Display Terminal (ODT) 1-6
OPTIONAL 4-79
Optional Items E-2
Options, How to Set 3-7
Organization of Manual xix
OU (Output Unit) 2-4, 2-7, 2-9
OU (Specify Output Device) 5-94
OUT 5-132
OUTPUT 4-80, 4-108
Output Message Syntax 7-1

INDEX (CONT)

Output Serial Files C-3
 output-unit-mnemonic 5-97
 OVERLAY.COUNTER A-8
 OVERLAY.RATE 5-80
 OVERLAY.TARGET A-8
 OVERRIDE 4-148
 PACK 5-40, 5-107, 5-141
 PACK <family-name> 5-42, 5-53, 5-64, 5-66,
 5-92, 5-94, 5-104, 5-120,
 5-130

 PACK.ID 4-81
 PAGE.SIZE 4-82
 PAN 5-166
 PAPER.TAPE.PUNCH 4-83
 PAPER.TAPE.READER 4-84
 PASSWORD 5-95
 password 5-158, D-1
 PB (Print Backup) 2-9
 PB (Print/Punch Backup) 5-96
 PBD 3-11
 PBT 3-11
 PCH/= 5-13, 5-99, 5-112, 5-118
 PD (Display Directory) 5-101
 PD (Print Directory) 2-3, 2-5
 Perform the CM(s) 3-16
 Performing Hexadecimal Conversion and
 Calculations 2-8
 PF (Print Fetch) 2-3, 2-5, 5-103
 PG (Purge) 2-5, 2-7, 5-104
 PM (Print Memory Dump) 2-6, 5-105
 PO (Power Off) 2-5, 2-7, 5-107
 PORT.FILE 4-85
 PORT.KEY 4-86
 PP (Processor Priority) 2-3, 2-8, 5-108
 PR (Change Priority) 5-109
 PR (Priority Change) 2-8
 PR (Priority) 2-3
 PRINTER 4-87
 PRIORITY 4-150
 Priority Memory Management A-9
 PRIVATE 4-107, 5-132
 PRN/= 5-13, 5-99, 5-112, 5-118
 PROCESSOR.PRIORITY 4-152
 Program Switch 2 6-1
 Program Switch 3 6-2
 Program Switch 7 6-2
 program-control-instruction 4-1 thru 4-5,
 4-7, 4-8, 4-10,
 4-13, 4-17, 4-19,

INDEX (CONT)

4-133 thru 4-141,
 4-143 thru 4-147,
 4-149, 4-150,
 4-153 thru 4-155,
 4-159, 4-161,
 4-164, 4-166,
 4-168, 4-169, 4-171,
 4-173 thru 4-175
 program-identifier 5-110
 program-name 1-8, 4-1 thru 4-6, 4-13, 4-17,
 4-18, 4-35, 4-133 thru 4-141,
 4-143 thru 4-146, 4-148, 4-150,
 4-152, 4-154, 4-155, 4-159,
 4-161, 4-164, 4-166, 4-168,
 4-169, 4-173, 4-174, 4-175,
 5-43
 program-name1 4-2, 4-8, 4-171
 program-name2 4-2, 4-8, 4-171
 PROTECTED 4-88, 4-154
 PROTECTED FILES C-6
 PROTECTION 4-88
 PROTOCOL 4-89
 PRT/= 5-13, 5-99, 5-112, 5-118
 PSEUDO 4-90
 PSR 5-92
 PSR.DECK 4-49
 PUBLIC 4-107, 5-132
 Punched Cards 1-5
 PURGE 5-20
 Purpose of Manual xix
 PV (Pack Override) 2-5
 Q.FAMILY.SIZE 4-91
 Q.MAX.MESSAGES 4-92
 QF (Query File) 2-3, 2-5, 5-110
 QP (Query Program) 2-1, 2-3, 5-111
 quartal-string 5-23, 5-151
 QUEUE 4-93
 QUIT Command 6-12
 Railroad Diagrams E-1
 RANDOM 4-94
 Random Files C-4
 RANGE 5-99
 RB (Remove Backup Files) 2-10, 5-112
 RC (Recover Data Base) 2-5
 RC (Recover Database) 5-114
 RD (Remove Decks) 2-10
 RD (Remove Pseudo Card Files) 5-115
 READER.PUNCH.PRINTER 4-97
 READER.SORTER 4-98

INDEX (CONT)

READER.SORTER.STATIONS 4-99
READER.96 4-100
RECORD 5-99
RECORD.SIZE.IN.BYTES 4-96
RECORDS.PER.BLOCK 4-95
REEL.NUMBER 4-101
Related Documentation xx
RELATIVE 4-49
RELEASE LINE Command 6-14
RELOAD LINE Command 6-13
REMOTE 4-102
remote-file-number 6-18
REMOVE 5-116
Remove Old Software 3-17
Remove Old System Files 3-15
REPETITIONS 4-103
Required Items E-2
RESET 5-40, 5-135, 6-7, 6-16
Responding to Program Messages 2-2
RESTART LINE 6-15
REVERSE 4-104
REWIND 4-105
RF (Remove Backup Files) 2-10, 5-118
RL (Relabel User Disk) 5-120
RL (Relabel) 2-5
RM (Remove Duplicate Disk File) 5-121
RM (Remove) 2-4
RMOV 3-11
RMSG 3-11
RN (Assign Pseudo Readers) 5-122
RN (Run Pseudo Readers) 2-10
RO (Reset Option) 2-6, 5-123
RP (Ready and Purge) 2-7, 5-124
RPG File Declarations C-10
RR 4-155
RS (Remove from Schedule) 2-9
RS (Remove Jobs from Schedule) 5-125
RS System Command 1-3
RT (Remove Multipack File Table Entry) 2-6, 5-127
RUN 4-157
RY (Ready Peripheral) 5-128
RY (Ready) 2-7
SAMPLING.CLOCK A-8
SAMPLING.INTERVAL 5-80, A-8
SAVE 4-7, 4-88, 4-106, 5-99
SB (Seconds Before Decay) 2-8
SB (Seconds Before decay) 5-129
SCHEDULE.PRIORITY 4-159
SCHM 3-11

INDEX (CONT)

SD (Assign Additional System Drives) 5-130
SD (System Disk) 2-11
SDL/UPL File Declarations C-9
SE (Enable Console Switches) 2-11
SE (Switch Enable) 5-131
SEC 5-77
second-name 5-58, 5-97
seconds 4-140
SECONDS.BEFORE.DECAY 4-161
SECURITY 5-132
SECURITYTYPE 4-107
SECURITYUSE 4-108
SEND.ALL.ATTRIBUTES 4-109
SEQUENTIAL 4-110
SERIAL 4-111
Serial Files C-2
serial-number 5-104, 5-130, 5-137, 5-138
SET 5-135, 6-7, 6-16
Setting the Operating System Options 3-7
SIMPLE.HEADERS 4-112
SINGLE 5-99
SIZE 5-62
SL (Set Log) 2-6, 2-10
SL (Set LOG) 5-133
SM (Set Database Parameters) 2-6, 5-134
SM (Set DB Parameters) 2-3
SN (Assign a Tape Serial Number) 5-137
SN (Serial Number) 2-7
SNL (Assign a Tape Serial Number and Lock) 5-138
SO (Set Option) 2-6, 5-139
software-mnemonic 5-20
Source Program Specifications C-7
Sources of Control Instructions 1-5
SP (Change Schedule Priority) 5-140
SP (Schedule Priority) 2-9
SP System Command 1-3
SPECIAL.FORMS 4-113
Specification Of Activities To Be Displayed B-3
Specification of I/O Activity Monitoring B-4
Specification of Overlay Activity Monitoring B-6
Specification of Processor Activity Monitoring B-5
SQ (Squash Disk) 5-141
SQ (Squash) 2-6
SQRM 3-12
ST (Stop) 2-8
ST (Suspend Processing) 5-143
Stand-alone Utility Cassettes (SDL Utility Programs) 3-2
STANDBY Command 6-16

INDEX (CONT)

START 4-158, 5-144
START (Start a WFL Job) 2-11
starting-record 5-97
STATION ALL 6-18
station-name 4-114, 6-4
STATIONS 4-114
STATISTICS 5-135
STATISTICS Command 6-17
STATUS 5-136
Status Checking 2-2
STATUS Command 6-18
STREAM 4-163
string 4-56, 4-70
string-1 5-98
string-2 5-98
string-3 5-98
Summary of Switch Specifications B-3
SUP 5-62
SUPPRESS-WARNINGS or SUPPRESS 5-136
SUS 5-77
SV (Save Peripheral Unit) 5-145
SV (Save) 2-7
SW (Set Switch) 5-146
SW (Set Switches) 2-8
SWITCH 4-164, 6-17
switch-number 4-164, 5-146
SY (System Program Pack) 5-147
SYMBOLIC.QUEUE.NAME 4-166
SYNCPPOINT 5-136
SYNTAX 4-7, 5-144
Syntax Diagrams 5-1
SYSTEM 5-141
System Cassettes, How to Create 3-1
SYSTEM CONSOLE B-1
System Description 1-1
SYSTEM disk 1-8
System Mix 1-4
system-command-in-WFL 5-160
system-initialization-file-name 5-87
SYSTEM/COPY and SDL2 Intrinsic File 3-16
System, How to clear/start 3-6
System, How to Coldstart 3-3
System, How to Initialize 3-3
System, How to Set Options 3-7
TABS 5-62, 5-71, 5-72
TAPE 4-115
TAPE.NRZ 4-116
TAPE.PE 4-117
TAPE.7 4-118

INDEX (CONT)

TAPE.9 4-119
Task Schedules and the System Mix 1-2
TD (Time and Date) 2-1, 2-3, 5-149
TEMPORARY 4-88
Temporary Operating Environment Changes 3-6
TERM 3-12
TERMINATE 4-167
text 6-8
TG (Trace Gismo) 5-150
The Master Control Program (MCP) 1-1
THEN 4-168
THR 3-12
THRASH 5-81
Thrashing A-2
Thrashing Detection A-8
THRASHING.SENSITIVITY 5-81
TI (Time In) 2-1
TI (Time Interrogate) 2-3
TI (Time Interrogation) 5-152
TIME 3-12, 4-169, 5-62
TITLE 4-120, 5-43
title-syntax 4-120
TL (Transfer Log) 2-10
TL (Transfer LOG) 5-153
TO (Display Options) 5-154
TO (Type Options) 2-3, 2-6
TR (Time Change) 5-155
TR (Time Reset) 2-1
train-identification 5-75
TRANSLATE 4-121
TRANSLATE.FILE.NAME 4-122
translate-file-title 4-122
TRMD 3-12
TS (Test Switches) 2-1, 2-4, 5-156
U/= 5-166
UL (Assign Unlabeled File) 5-157
UL (Unlabeled File) 2-4
UL (Unlabeled) 2-7
UNCONDITIONAL.EXECUTION 4-171
UNFREEZE 4-173
UNIT 5-40
UNIT.NAME 4-123
unit-mnemonic 1-8, 4-123, 5-3, 5-7, 5-19,
5-42, 5-53, 5-75, 5-92, 5-94,
5-98, 5-104, 5-107, 5-120,
5-124, 5-128, 5-130, 5-137,
5-138, 5-141, 5-145, 5-157
unit-type-code 5-92
UNLABELED 4-124

INDEX (CONT)

UNOVERRIDE 4-174
UNS 5-62
UPPER 5-99
UPPER.MARGIN 4-125
US 5-166
Use Experimental Entries 3-14
USER Command 5-158, D-1
user disk 1-8
USER.BACKUP.NAME 4-126
usercode 5-98, 5-158, D-1
Using the COLDSTART/DISK Cassette 3-4
Using the COLDSTART/TAPE Cassette 3-3
Using the Data Communications ODT as
a Terminal 6-21
Variable and Fixed Lamp Displays B-7
VARIABLE.BLOCK 4-127
VIRTUAL.DISK 4-175
VLCP 3-12, B-2
VLIO 3-12, B-2
WAITING 5-162
Waiting Schedule 1-3
WD (Display MCP Date) 5-159
WD (What Date) 2-1, 2-4
WFL 3-13
WFL (Work Flow Language) Command 5-160
WITH.INTERPRET 4-128
WITH.PRINT 4-129
WITH.PUNCH 4-130
WITH.STACKERS 4-131
WM (Display Current MCP and Interpreter) 5-161
WM (What MCP) 2-2, 2-4
Work Flow Language (WFL) Command 2-11
WORK.FILE 4-132
Working Set A-2
WS (Display Schedule) 5-162
WS (What Schedule) 2-4, 2-9
WS System Command 1-3
WT (Display MCP Time) 5-163
WT (What Time) 2-1, 2-4
WW (List Contents of the Name Table) 5-164
WW (Name Table) 2-4
WY (Mix and Status Interrogation) 5-167
WY (WHY) 2-2
WY (Why) 2-4
x 5-64, 5-66
XC (Lockout Disk) 2-6
XC (Remove Segments Temporarily) 5-168
XD (Lockout Disk) 2-6
XD (Remove Segments Permanently) 5-169

INDEX (CONT)

ZIP 3-13

ZIP Statement 1-6

10-digit-hexadecimal-number 4-165, 5-146

