

Burroughs
B 5500/B 5700
ELECTRONIC INFORMATION
PROCESSING SYSTEMS
OPERATION MANUAL



Burroughs Corporation
Detroit, Michigan 48232

\$15.00

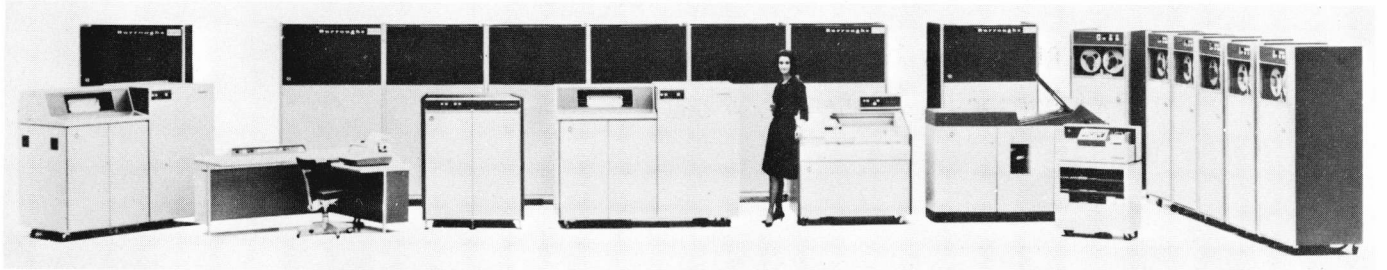
COPYRIGHT © 1966, 1964, 1963 BURROUGHS CORPORATION
AA 820300, AA 745636, AA 662176, AA 662175, AA 619948

The revised edition of this manual, dated 9-68, incorporated the following PCN's:

1024916-001 (Mar. 1, 1968)
1024916-002 (Mar. 15, 1968)
1024916-003 (May 15, 1968)
1024916-004 (Jun. 6, 1968)
1024916-005 (Jul. 15, 1968)
1024916-006 (Aug. 29, 1968)

This reprint includes the information released under the following PCN's:

1024916-007 (Feb. 21, 1969)
1024916-008 (Mar. 28, 1969)
1024916-009 (May 1, 1969)
1024916-010 (May 26, 1969)
1024916-011 (Sep. 5, 1969)



Burroughs B 5500 Electronic Information Processing System

TABLE OF CONTENTS

SECTION	TITLE	PAGE
	INTRODUCTION	xvii
1	SYSTEM DESCRIPTION	1-1
	General.	1-1
	Functional Description	1-2
	System Design.	1-2
	Data Communications System	1-3
2	SYSTEM EQUIPMENT	2-1
	General.	2-1
	Operator Console	2-1
	Control Panel.	2-2
	Supervisory Printer.	2-4A
	Functional Characteristics	2-4A
	Control Panel.	2-5
	B 5005 Basic Auxiliary Memory Subsystem.	2-8
	Control Panel.	2-8A
	B 122 Card Reader.	2-8B
	Functional Characteristics	2-8C
	Control Panel.	2-9
	Operating Procedures	2-11
	Not Ready Conditions	2-12
	Card Jam	2-13
	Stacker Full	2-14
	Cover Not in Place	2-15
	Empty Hopper	2-15
	STOP Switch Pressed.	2-15
	FEED CHECK Indicator Lit	2-15
	READ CHECK Indicator Lit	2-15
	Operator Maintenance	2-16
	B 123 Card Reader.	2-17
	Functional Characteristics	2-17
	Control Panel.	2-19

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	Operating Procedures	2-21
	Not Ready Conditions	2-26
	Read Check Condition	2-27
	Feed Check Condition	2-27
	Card Jam ^{INSTRUCTION}	2-28
	Stacker Full	2-29
	Cover Not in Place	2-29
	Empty Hopper	2-29
	STOP Switch Pressed.	2-30
	READ CHECK Indicator Lit	2-30
	FEED CHECK Indicator Lit	2-30
	Operator Maintenance	2-32
B 9111 and B 9112	Card Readers	2-33
B 9111	Card Reader	2-33
	Functional Characteristics	2-34
	Control Panel.	2-34
	Operating Procedures	2-34
	Not Ready Conditions	2-34A
	Feed Error Condition	2-34A
	Card Jam	2-34A
	Operator Maintenance	2-34A
B 9210	Card Punch.	2-34A
	Functional Characteristics	2-34A
	Control Panel.	2-34D
	Operating Procedures	2-36
	Unloading Cards.	2-37
	Not Ready Conditions	2-37
	FEED CHECK Indicator Lit	2-38
	STOP Switch Pressed.	2-43
	Empty Hopper	2-43
	Cover Opened	2-43

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	Card Not at Ready Station.	2-43
	Punch Die Not in Place	2-43
	Card Not at Read Station	2-43
	PUNCH CHECK Indicator Lit.	2-43
	Stacker Full	2-44
	Operator Maintenance	2-44
B 9211	Card Punch.	2-45
	Functional Characteristics	2-46
	Control Panel.	2-47
	Operating Procedures	2-49
	Unloading Cards.	2-51
	Not Ready Conditions	2-52
	FEED CHECK Indicator Lit	2-53
	STOP Switch Pressed.	2-56
	Empty Hopper	2-56
	Feed Roll Block Not Locked	2-56
	Punch Block Not Locked	2-56
	Card Not at Prepunch Station	2-57
	Primary, Error, or Auxiliary Stacker Full	2-58
	Covers Not in Place.	2-58
	PUNCH CHECK Indicator Lit.	2-58
	Operator Maintenance	2-58
B 9212-1/B 9213-1	Card Punches	2-59
B 9213-1	Card Punch.	2-59
	Functional Characteristics	2-60
	Control Panel.	2-61
	Operating Procedures	2-62A
	Unloading Cards.	2-62B
	Not Ready Conditions	2-62B
	Empty Hopper	2-62C
	Full Stacker	2-62C
	Failure to Feed.	2-62C

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	Pressing STOP Switch	2-62D
	Jam Condition.	2-62D
	Punch Mechanism Not Locked	2-62E
	Feed Roll Block Not Locked	2-62E
	Cover Not Closed	2-62F
	PUNCH CHK Indicator On	2-62F
	Operator Maintenance	2-62F
B 9240/B 9241	Line Printers.	2-62G
B 9240	Line Printer.	2-62G
	Functional Characteristics	2-62G
	Control Panel.	2-62H
	Forms Handling	2-63
	Tape Punching.	2-64
	Operating Procedures	2-66
	Changing the Ribbon.	2-69
	Inserting the Carriage Control Tape.	2-74
	Tape and Forms Registration.	2-75
	Not Ready Conditions	2-76
	END OF PAPER Indicator Lit	2-76
	Print Drum Not in Position	2-77
	Line Selection Knob in N Position.	2-77
	Paper Slews for More than One Second	2-77
	STOP Switch Pressed.	2-77
	Operator Maintenance	2-77
B 9242 and B 9243	Series Line Printers	2-78
B 9242-4	Line Printer.	2-78A
	Functional Characteristics	2-78A
	Control Panel.	2-78B
	Forms Handling	2-78C
	Tape Punching.	2-78C
	Operating Procedures	2-78D
	Changing the Ribbon.	2-78G

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	Inserting the Carriage Control Tape.	2-78J
	Tape and Forms Registration.	2-78K
	Not Ready Conditions	2-78K
	END OF PAPER Indicator Lit	2-78L
	Print Drum Not In Position	2-78L
	Paper Slew for More Than One Second.	2-78L
	START/STOP Switch Pressed.	2-78L
	Operator Maintenance	2-78L
B 9120	Paper Tape Reader	2-78M
	Functional Characteristics	2-80
	Channel Select Plugboard	2-80
B 9926	Input Code Translator	2-81
	Control Panel.	2-84
	Operating Procedures	2-87
	Stopping Tape Movement	2-90
	Unloading Paper Tape	2-91
	Applying Adhesive Opaque Strips.	2-91
	Operator Maintenance	2-92
B 9220	Paper Tape Punch.	2-93
	Functional Characteristics	2-94
	Channel Select Plugboard	2-95
B 9928	Code Translator	2-95
	Control Panel.	2-100
	Loading Paper Tape	2-102
	Unloading Tape	2-105
	Rewinding Tape	2-105
	Chad Receptacle.	2-106
	Splicing Paper Tape.	2-106
	Operator Maintenance	2-107
B 9391/B 9394-1/B 9396/B 9396-1	Magnetic Tape Units.	2-108
B 9396	Magnetic Tape Unit.	2-108
	Functional Characteristics	2-109

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	Control Panel	2-110
	Loading the Supply Reel	2-112
	Unloading the Supply Reel	2-116
	Loading the Take-Up Reel	2-117
	Unloading the Take-Up Reel	2-118
	Rewinding	2-118
	Attaching Leaders	2-118
	Splicing Magnetic Tape	2-120
	Operator Maintenance	2-121
	Magnetic Tape Care	2-122
	Magnetic Tape Storage	2-122
	Magnetic Tape Handling	2-123
	Magnetic Tape Loading	2-123
	Magnetic Tape Library Procedures	2-124
B 9410	Peripheral Switching Unit	2-124
	Control Panel	2-126
	Disk File System	2-128
	Functional Description	2-128
B 450	Basic Disk File/Data Communications Cabinet	2-129
B 5374-3	Basic Disk File/Data Transmission Terminal Unit Cabinet	2-129
B 5374-4	Disk File Expanded Control	2-129
B 5374-5	Disk File Control Unit	2-130
B 9373	Disk File Electronics Unit	2-132
	Control Panel	2-132
	Disk Lockout Switches	2-134
B 9374-1	Systems Memory Storage Module	2-134
B 5376	File Protect Memory (Shared Disk System)	2-136
	Data Communications Systems	2-138
	Data Communications System I	2-139
B 5352-2	Data Transmission Control Unit (DTCU)	2-139

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
2 (cont)	B 5352-3 Data Transmission Terminal Unit (DTTU)	2-140
	Buffer Conditions	2-141
	B 5350 Data Communications Processor (System II)	2-143
	Control Panel	2-145
3	LOADING AND MAINTAINING THE SYSTEM	3-1
	General	3-1
	Systems Material	3-1
	SYMBOL Tape	3-1
	Basic Change Deck	3-6
	Merging Patches	3-6
	Compiling Source	3-7
	Examples	3-7
	SYSTEM Tape	3-14
	Card Load Select Programs	3-14
	ESPOL Loader Program	3-15
	Halt/Load Program	3-15
	Cold Start Program	3-15
	Cool Start Program	3-16
	Tape to Disk MCP Loader Program	3-16
	Disk to Disk MCP Loader Program	3-17
	Halt/Load Kernel Program	3-17
	Core to Tape Dump Program	3-17
	MCP Loader Decks	3-17
	Tape to Disk MCP Loader Deck	3-18
	Disk to Disk MCP Loader Deck	3-18
	System Loader Decks	3-19
	Cold Start Deck	3-19
	DRCTRYTP Card	3-20
	DIRECT Card	3-20

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
3 (cont)	ESU Card	3-21
	SYSTEMS Card	3-21
	FENCE Card	3-22
	DATE Card.	3-22
	FILE Card Group.	3-23
	OPTION Cards	3-26
	STOP Card.	3-35
	Cool Start Deck	3-35
	Control Cards for System Loading	3-36
	Disk Halt/Load Card.	3-37
	Control Cards Used to Load Compilers onto Disk.	3-37
	System Procedures	3-37
	System Start-Up Procedure.	3-38
	Loading the System from the System Tape	3-38
	Program Scheduling Information	3-39
	The Selection Algorithm.	3-40
	The Multiprocessing Factor	3-41
4	CONTROL INFORMATION.	4-1
	General	4-1
	Conventions.	4-1
	Definitions.	4-1
	Control Information Via Punched Cards.	4-8
	Control Cards	4-8A
	COMPILE Card	4-9
	Compile-and-Go Run.	4-9
	Compile-for-Library Run	4-9
	Compile-for-Syntax-Check Run.	4-10
	EXECUTE Card	4-10
	REMOVE Card.	4-11
	DUMP Card - UNLOAD Card.	4-12
	LOAD Card - ADD Card	4-15
	CHANGE Card.	4-16A

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
5 (cont)	Loading a Control Deck File onto Disk	5-1
	Card Reader CONTROL DECK File.	5-2
	Magnetic Tape CONTROL DECK File.	5-2
	Pseudo Decks on Disk	5-2
	Removing Pseudo Decks from Disk.	5-3
	Copying a Control Deck onto Tape	5-3
	Calling the LDCNTRL/DISK Program Out for Execution.	5-3
	Parity on a Control Deck Magnetic Tape File	5-4
	Pseudo Card Readers and the Use of Pseudo Card Decks.	5-4
	The RN Message to Turn On Pseudo Card Readers.	5-4
	The RN Message to Turn Off Pseudo Card Readers.	5-5
	Removing Decks from Pseudo Card Readers	5-5
	Handling of Control Card Errors in Pseudo Card Decks	5-5
	Symbolic Library File on Disk.	5-5
	Control Card Syntax	5-6
	Semantics.	5-8
	Maintenance Function Examples.	5-10
	End of Job and Error Messages	5-18
	Setup	5-18
	Copying Symbolic Library Tapes onto Disk	5-19
	Log Maintenance.	5-19
	Log Entry Specifications.	5-19
	Code Word.	5-20
	Control Card Information	5-20
	Compiler and Object Program Information.	5-22
	Special Records and Log Initialization.	5-26A
	Record Zero.	5-26A
	Record n + 1	5-26B

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
	Initializing the Log	5-26B
	Remote Log Specifications.	5-26B
	Log Entry Specifications	5-26B
	Type 1 Log-Out Entry	5-27
	Type 2 Log-In Entry.	5-27
	Type 3 Control Card Entry.	5-27
	Type 4 Control Card Entry.	5-28
	Type 5 Job Statistics.	5-28
	Creation of Remote Log Entries	5-29
	File Maintenance Procedures.	5-31
	Statistics Log	5-32
	General Characteristics	5-32
	Time Sharing.	5-34
	Standard System	5-34
	Operation	5-34
	File Descriptions	5-35
	System Statistics File	5-35
	Statistics Log File.	5-38
	Time-Sharing Log Additions	5-39
	Disk Directory	5-41
	Printer Backup Information	5-44
	Format of a PBT	5-44
	Format of Blocks on a PB File	5-45
	Format of Records on a PB File.	5-45
	Format of Printer Backup File on Disk	5-46
	File Opening Action	5-47
	Special Forms	5-47
	Closing a Print File on Disk.	5-48
	Logging of PB Files.	5-48
APPENDIX A	- Character Representation.	A-1
APPENDIX B	- Identifiers	B-1
APPENDIX C	- Messages.	C-1
APPENDIX D	- The Remote SPO Station Facility	D-1
APPENDIX E	- File Security System.	E-1

TABLE OF CONTENTS (cont)

SECTION	TITLE	PAGE
APPENDIX F	- Extended ALGOL Syntactical Error Messages	F-1
APPENDIX G	- COBOL Compiler Error and Diagnostic Messages.	G-1
APPENDIX H	- FORTRAN Compiler Error Messages	H-1
APPENDIX I	- FORTRAN Translator Error Messages	I-1
APPENDIX J	- Compatible ALGOL Compiler Error Messages.	J-1
APPENDIX K	- The Breakout Process.	K-1
INDEX.		one

LIST OF ILLUSTRATIONS

FIGURE	TITLE	PAGE
1-1	Functions of a Data Processing System.	1-1
1-2	Typical Systems Configuration.	1-3
2-1	Operator Console	2-2
2-2	Operator Console Control Panel	2-2
2-3	Supervisory Printer.	2-5
2-4	Supervisory Printer and Keyboard Controls.	2-6
2-4A	B 5005 Basic Auxiliary Memory Subsystem.	2-8
2-4B	B 5005 Basic Auxiliary Memory Subsystem Control Panel.	2-8A
2-5	B 122 Card Reader.	2-8C
2-6	B 122 Card Reader Control Panel.	2-9
2-7	B 122 Card Reader Read Mechanism	2-13
2-8	B 122 Card Reader Read Head Removed.	2-14
2-9	B 123 Card Reader.	2-17
2-10	B 123 Card Reader Control Panel.	2-19
2-11	Stacker End Plate Assembly	2-21
2-12	Adjusting Stacker Front Wall	2-22
2-13	Loading Cards.	2-24
2-14	B 9111 Card Reader	2-33
2-14A	B 9111 Card Reader Control Panel	2-34
2-14B	B 9111 Card Reader Guide Plate Area.	2-34B
2-15	B 9210 Card Punch.	2-34B
2-16	B 9210 Card Punch Control Panel.	2-34D

LIST OF ILLUSTRATIONS (cont)

FIGURE	TITLE	PAGE
2-17	Loading Blank Cards.	2-37
2-18	Unloading Punched Cards.	2-38
2-19	Card Punch Assembly, Top View.	2-39
2-20	Removing the Chip Box.	2-40
2-21	Chip Box Area.	2-41
2-22	B 9211 Card Punch.	2-45
2-23	B 9211 Card Punch Control Panel.	2-47
2-24	Loading Cards.	2-50
2-25	Removing Cards from Error Stacker.	2-51
2-26	Removing Cards from Normal Stacker	2-52
2-27	B 9211 Card Punch, Left Side Open.	2-54
2-28	B 9211 Card Punch, Right Side Open	2-55
2-28A	B 9213-1 Card Punch.	2-60
2-28B	B 9213-1 Card Punch Control Panel.	2-62
2-28C	B 9213-1 Card Punch, Top Cover Raised.	2-62D
2-29	B 9240 Line Printer.	2-62G
2-30	B 9240 Line Printer Control Panel.	2-62I
2-31	Carriage Control Tape Punch.	2-65
2-32	B 9240 Line Printer Component Layout	2-67
2-33	Line Printer Paper Guide, Rear View.	2-70
2-34	Removing the Printer Ribbon.	2-71
2-35	Side View of Print Mechanism	2-72
2-36	Ribbon in Proper Position.	2-73
2-37	Removing Ribbon from Tracking Device	2-74
2-38	Carriage Control Tape Mechanism.	2-75
2-38A	B 9242-4 Line Printer.	2-78A
2-38B	B 9242-4 Line Printer Control Panel.	2-78B
2-38C	B 9242-4 Line Printer Component Layout.	2-78D
2-39	B 9120 Paper Tape Reader	2-79
2-40	Channel Select Plugboard Wiring.	2-80
2-41	Plugboard Layout	2-82
2-42	B 9120 Paper Tape Reader Control Panel	2-84
2-43	Threaded Paper Tape.	2-89

LIST OF ILLUSTRATIONS (cont)

FIGURE	TITLE	PAGE
2-44	Paper Tape in Operating Position	2-90
2-45	Location of Opaque Strip on Paper Tape	2-92
2-46	B 9220 Paper Tape Punch.	2-94
2-47	Channel Select Plugboard	2-96
2-48	Plugboard Layout	2-97
2-49	B 9220 Paper Tape Punch Control Panel.	2-101
2-50	B 9220 Paper Tape Punch Transport.	2-104
2-51	B 9396 Magnetic Tape Unit.	2-109
2-52	Magnetic Tape Unit Control Panel	2-111
2-53	Tape Follower Arm and Clamp.	2-113
2-54	Magnetic Tape Reel with Write Ring	2-114
2-55	Mounting the Supply Reel	2-114
2-56	Connecting the Tape Leader	2-115
2-57	Cutting Ends of Magnetic Tape.	2-119
2-58	Applying Adhesive Tape to Magnetic Tape Ends . . .	2-120
2-59	Location of BOT and EOT Markers.	2-120
2-59A	B 9410 Peripheral Switching Unit	2-125
2-59B	B 9410 Peripheral Switching Unit Control Panel . .	2-126
2-60	B 450 Disk File/Data Communications Cabinet. . . .	2-131
2-61	B 9373 Disk File Electronics Unit.	2-132
2-62	B 9373 Disk File Electronics Unit Control Panel. .	2-133
2-63	Disk Lockout Switches.	2-135
2-64	B 9374-1 Systems Memory Storage Module	2-135
2-65	B 5350 Data Communications Processor	2-143
5-1	Log Entry Formats.	5-21
5-2	Format of General Program Information.	5-22
5-3	Format of One File-Information Record.	5-24

LIST OF TABLES

TABLE	TITLE	PAGE
1-1	System Configuration Chart	1-4
2-1	Operator Console Switches and Indicators	2-2

LIST OF TABLES (cont)

TABLE	TITLE	PAGE
2-2	Supervisory Printer Switches and Indicators.	2-6
2-2A	B 5005 Basic Auxiliary Memory Subsystem Control Panel Switches and Indicators.	2-8A
2-3	B 122 Card Reader Control Panel Switches and Indicators	2-9
2-4	B 123 Card Reader Control Panel Switches and Indicators	2-19
2-5	B 9210 Card Punch Control Panel Switches and Indicators	2-34D
2-6	B 9211 Card Punch Control Panel Switches and Indicators	2-47
2-6A	B 9213-1 Card Punch Control Panel Switches and Indicators	2-62
2-7	B 9240 Line Printer Control Panel Switches and Indicators	2-62J
2-7A	B 9242-4 Line Printer Control Panel Switches and Indicators	2-78B
2-8	B 9120 Paper Tape Reader Control Panel Switches and Indicators.	2-85
2-9	B 9220 Paper Tape Punch Control Panel Switches and Indicators.	2-101
2-10	Magnetic Tape Unit Control Panel Switches and Indicators	2-111
2-10A	B 9410 Peripheral Switching Unit Control Panel Switches and Indicators.	2-127
2-11	B 9373 Disk File Electronics Unit Control Panel Switches and Indicators.	2-133
3-1	SYMBOL/SYSTEM File Relationships	3-1

INTRODUCTION

The productivity of a computer facility is largely dependent on an operator's experience and knowledge of the hardware. When the programs produced for the installation have been refined and are ready for use, the results obtained are largely controlled by the operator. Therefore, some concept of the B 5500 System's logic and a thorough knowledge of the hardware are important in order for the operator to utilize the equipment effectively.

In preparing this manual for the B 5500 Information Processing System, it was necessary to make some assumptions which affect its content. A presumption was made that the reader is familiar with the components of the system and has some perception of their functions. Without this presumption, it would be difficult to prepare a manual and still remain within reasonable limitations.

This manual is divided into the following five sections, and provides a complete reference and operating guide, thus enabling personnel to perform their duties efficiently on the B 5500 System.

- SYSTEM DESCRIPTION Describes the B 5500 System configuration and its functional capabilities.
- SYSTEM EQUIPMENT Depicts the units of peripheral equipment that make up a B 5500 System and the necessary procedures for placing them in operation.
- LOADING AND MAINTAINING THE SYSTEM Presents the procedures for starting the system and introducing jobs both for initial start-up and in-process conditions.
- CONTROL INFORMATION. Describes in detail the various types of cards which supply control information to the B 5500 System.

UTILITY ROUTINES Presents and explains the routines in the programing system that are designed to facilitate the job of the programmer and operator.

In addition, nine appendices supplement the subjects dealt with in the text.

It should be understood that the information in this manual has been acquired by Sales Technical Services from actual operating experience. Each installation may encounter unique conditions in its operations which may not be covered in this manual. Therefore, the assistance of Burroughs Corporation is provided for any phase of the operation by the District Sales Technical and Field Engineering staffs.

SECTION 3
LOADING AND MAINTAINING THE SYSTEM

GENERAL.

This section describes the materials and procedures for loading and maintaining the system.

SYSTEMS MATERIAL.

The following materials are required for loading and maintaining the system. The items are described in more detail in subsequent paragraphs.

- a. SYMBOL Tape(s).
- b. Basic change decks.
- c. SYSTEM Tape(s).
- d. Card Load Select Programs.
- e. MCP Loader Decks.
- f. System Loader Decks.
- g. Disk Halt/Load Card.
- h. Control cards for system loading.

SYMBOL TAPE.

The SYMBOL Tape is a multifile tape; i.e., each SYMBOL Tape can contain one or more files. Each file on the SYMBOL Tape is the symbolic (source) program for a specific system (object) program. Table 3-1 shows the relationship between symbolic and system programs. Updated SYMBOL Tapes are released to installations when the MARK level (systems software level) is changed; this allows all previous SYMBOL Tapes, basic change decks, and patches to be discarded.

Table 3-1
SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
SYMBOL/DCEPSY	ESPOL	MCP/DISK	Data Communications Master Control Program

Table 3-1 (cont)
 SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
SYMBOL/INTRNSY	ESPOL	INT/DISK	Data communications intrinsic routines for MCP
SYMBOL/DUMPTAP	ESPOL		Routine which generates machine language core dump deck
SYMBOL/COOLSY	ESPOL		Routine which generates Cold Start or Cool Start Deck
SYMBOL/TAPEDSK	ESPOL		Routine which generates MCP Tape to Disk Loader Deck
SYMBOL/DSKDSK	ESPOL		Routine which generates MCP Disk to Disk Loader Deck
SYMBOL/KERNAL	ESPOL		Routine which generates MCP Directory Search and Disk to Core Loader Deck
SYMBOL/PMERG	XALGOL	PATCH/MERGE	Program to edit and merge patches to update software
SYMBOL/ESPOL	ALGOL	ESPOL/DISK	ESPOL Compiler Programming System
SYMBOL/XALGOL	ALGOL	XALGOL/DISK	Compatible ALGOL Compiler Programming System
SYMBOL/ALGOL	ALGOL	ALGOL/DISK	ALGOL Compiler Programming System
SYMBOL/TSPOL	ALGOL	TSPOL/DISK	TSPOL Compiler Programming System
SYMBOL/BASIC	ALGOL	BASIC/DISK	BASIC Compiler Programming System
SYMBOL/COBOL	ALGOL	COBOL/DISK	COBOL Compiler Programming System
SYMBOL/COBOL68	ALGOL	COBOL68/DISK	COBOL (CODASYL-68) Compiler Programming System

Table 3-1 (cont)
 SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
SYMBOL/FORTRAN	ALGOL	FORTRAN/DISK	FORTRAN Compiler Programming System
SYMBOL/FORTLTR	ALGOL	FORTRAN/TRANS	FORTRAN Translator Conversion System
SYMBOL/CFILTER	ALGOL	CFILTER/DISK	COBOL Filter Conversion System
SYMBOL/AFILTER	ALGOL	AFILTER/DISK	ALGOL Filter Conversion System
SYMBOL/DCFILL	ALGOL	DCFILL/PRT	Data communications program to create STUFF Disk File for use with DUMP/ANALYZE
SYMBOL/DUMPANL	ALGOL	DUMP/ANALYZE	Data communications program to analyze and print memory dump
SYMBOL/MAKCAST	ALGOL	MAKCAST/DISK	System to create and maintain symbolic library files
SYMBOL/TEXTED	ALGOL	TEXT/EDITOR	System to create and maintain symbolic disk files remotely
SYMBOL/INTERP	ALGOL	INTERP/DISK	System for rapid computational use of remote console
SYMBOL/UPDATE	ALGOL	UPDATE/USERS	Data communications program to create or update valid security file with MCP file security procedures
SYMBOL/LOGOUT	ALGOL	LOGOUT/DISK	Data communications program for analysis of standard log
		LDCNTRL/DISK	Object program to load decks into pseudo readers

Table 3-1 (cont)
SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
SYMBOL/DCLOGAN	ALGOL	REMOTE/LOGAN	Data communications program for analysis of remote log
SYMBOL/MASTER	ALGOL	MASTER/TEST	COBOL maintenance test routine
SYMBOL/CHECKAL	ALGOL	CHECKAL/ALGOL	ALGOL maintenance test routine
SYMBOL/TSSMCP	ESPOL	TSS/MCP	Time Sharing MCP
SYMBOL/INTRNSY	ESPOL	TSS/INT	Time sharing intrinsic routines for MCP
SYMBOL/TSFILL	TSPOL	TSFILL/PRT	Time sharing program to create STUFF Disk File for use with TSDUMP/ANALYZE
SYMBOL/TSDUMP	TSPOL	TSDUMP/ANALYZE	Time sharing program to analyze and print memory dump
SYMBOL/TAPDSK	ESPOL		Routine to generate Time Sharing MCP Tape to Disk Loader Deck
SOURCE/CANDE	TSPOL	CANDE/TSHARER	Time Sharing Program Command and Edit Data Communications Handler
USERSC/CANDE	TSPOL	USER/CANDE	Time sharing utility program
MESAGE/CANDE		MESAGE/CANDE	Time sharing utility program
MSLDSC/CANDE	TSPOL	MSGLDR/CANDE	Time sharing utility program
SYSDSK/CANDE	TSPOL	SYSDISK/MAKER	Time sharing utility program
		PRNPBT/DISK	Object program to print MCP printer backup tapes

Table 3-1 (cont)
SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
HARDSC/CANDE	TSPOL	HARD/CANDE	Time sharing utility program
LETRSC/CANDE	TSPOL	LETTER/CANDE	Time sharing utility program
PATCH/FIND	TSPOL	FIND/CANDE	Time sharing verb routine
PATCH/QUIKLST	TSPOL	QUIKLST/CANDE	Time sharing verb routine
PATCH/REPLACE	TSPOL	REPLACE/CANDE	Time sharing verb routine
SKEDSC/CANDE	TSPOL	SCHEDUL/CANDE	Time sharing verb routine
LOADSC/CANDE	TSPOL	LOAD/CANDE	Time sharing verb routine
LISTSC/CANDE	TSPOL	LIST/CANDE	Time sharing verb routine
MERGSC/CANDE	TSPOL	MERG/CANDE	Time sharing verb routine
RESEQSC/CANDE	TSPOL	RESEQ/CANDE	Time sharing verb routine
GARDSC/CANDE	TSPOL	GUARD/DISK	Time sharing verb routine
COPYSC/CANDE	TSPOL	COPY/CANDE	Time sharing verb routine
FILESC/CANDE	TSPOL	FILES/CANDE	Time sharing verb routine
DLETSC/CANDE	TSPOL	DELETE/CANDE	Time sharing verb routine
PAPRSC/CANDE	TSPOL	PAPER/CANDE	Time sharing verb routine

Table 3-1 (cont)
SYMBOL/SYSTEM File Relationships

SYMBOL File	Compiler	SYSTEM File	Program Function
APNDSC/CANDE	TSPOL	APPEND/CANDE	Time sharing verb routine
PLSTSC/CANDE	TSPOL	PLIST/CANDE	Time sharing verb routine
RSQBSC/CANDE	TSPOL	RESEQB/CANDE	Time sharing verb routine
SYMBSC/CANDE	TSPOL	SYMBOL/CANDE	Time sharing verb routine
PNCHSC/CANDE	TSPOL	PUNCH/CANDE	Time sharing verb routine
LFILSC/CANDE	TSPOL	LFILS/CANDE	Time sharing verb routine

BASIC CHANGE DECKS.

Basic change decks have three purposes:

- a. Execution of the PATCH/MERGE routine to analyze patch releases and to generate a newly sequenced patch deck in the form of a pseudo reader file.
- b. Incorporation of the sequenced patch deck with the SYMBOL (source) Tape.
- c. Compilation of the SYMBOL Tape, with the patches, to create an object file.

MERGING PATCHES. The basic change deck setup for the analysis, substitution, sorting, and printing of patches by PATCH/MERGE is:

- a. ? EXECUTE PATCH/MERGE
- b. ? DATA CARD
- c. \$.<number of patch decks> PATCHES FOR
<multifile ID>.<mark level> CONFLICTS

- d. \$*COMPILE <object multifile ID>/<object file ID>
<compiler> LIBRARY
- e. \$*<compiler> FILE TAPE = SYMBOL/<source file ID>
- f. \$*FILE <input/output option> = <change option>
- g. \$*DATA CARD
- h. \$ <compiler option cards>
- i. \$#<patch deck serial number> FOR <multifile ID>.<mark
level number> <number of cards in this patch deck>
- j. <patch cards>
- k. ? END

If more than one release patch deck is run at the same time, the cards listed in i, j, and k above must be repeated for each deck.

COMPILING SOURCE. Once the PATCH/MERGE Operation has been completed, the compile deck with the edited patches may be executed by entering the message RN <number of pseudo readers>. This causes the edited patches to be merged with the SYMBOL <source program> Tape by the compiler, and an object file to be created. In most cases, this object file resides on disk; exceptions are those routines required to be in card-deck format, such as the Tape to Disk MCP Loader Program Deck. (It should be noted that these decks are produced with label cards which should be discarded.) In cases where the object disk file has been compiled under a name other than that in the relationship table, the name of the old version should be changed through use of the Change Control Card to avoid a duplicate library condition.

EXAMPLES. Basic change deck format examples for different conditions are given in the following paragraphs.

An example of a basic change deck format for the Master Control Program is as follows:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR MCP.XI CONFLICTS
$*EXECUTE ESPOL/DISK .COMPILE STANDARD MK XI MCP
$*PROCESS = 600; IO = 600
$*FILE STUFF = 0000000/MCP DISK
$*FILE TAPE = SYMBOL/DCESPSY
$*FILE NEWTAPE = MCP/SOURCE TAPE
$*FILE LINE = MCP/LISTING BACK UP DISK
$*DATA CARD
$$-CARDS NEEDED FOR MCP COMPILATION
$ SET BREAKOUT = TRUE
$ SET BREAKOUT = FALSE
$ SET DEBUGGING = FALSE
$ SET DEBUGGING = TRUE
$ SET DUMP = FALSE
$ SET DUMP = TRUE
$ SET DFX = FALSE
$ SET DFX = TRUE
$ SET INQUIRY = FALSE
$ SET INQUIRY = TRUE
$ SET DATACOM = FALSE
$ SET DATACOM = TRUE
$ SET DCSP0 = FALSE
$ SET DCSP0 = TRUE
$ SET DCLOG = FALSE
$ SET DCLOG = TRUE
$ SET CHECKLINK = FALSE
$ SET CHECKLINK = TRUE
$ SET DISKLOG = FALSE
$ SET DISKLOG = TRUE
$ TAPE LIST PRT STUFF NEW TAPE
.
.
.

```

NOTE

Since the last card for each option determines its setting, an option may be changed by changing its card position.

```

$#PATCH NUMBER 1 FOR MCP.XI CONTAINS n CARDS
<n sorted patches>

```

```
##PATCH NUMBER 2 FOR MCP.XI CONTAINS n CARDS
      <n sorted patches>
##PATCH NUMBER xx FOR MCP.XI CONTAINS n CARDS
      <n sorted patches>
? END
```

NOTE

When the MCP is compiled, it is necessary to execute the FILL/PRT Program which builds the MCP/PRT Disk File used by DUMP/ANALYZE. If the MCP is compiled in the manner shown, FILL/PRT should be executed as follows:

```
? EXECUTE FILL/PRT
? FILE MCP = 0000000/MCP DISK
? FILE INT = 0000000/INT DISK
? END
```

An example of a basic change deck format for INTRINSICS is as follows:

```
? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR INTRINSICS.XI CONFLICTS
$*EXECUTE ESPOL/DISK .COMPILE STANDARD INTRINSICS
$*FILE STUFF = 0000000/INT DISK
$*FILE NEWTAPE = INT/SOURCE TAPE
$*FILE DISK = INT/DISK
$*FILE TAPE = SYMBOL/INTRNSY
$*PROCESS = 60; IO = 60
$*DATA CARD
$-CARDS NEEDED FOR INTRINSICS COMPILATION
$ SET TIMESHARING = TRUE
$ SET TIMESHARING = FALSE
$ SET INQUIRY = TRUE
$ SET INQUIRY = FALSE
```

```

$ TAPE INTRINSIC LIST PRT STUFF
##PATCH NUMBER 1 FOR INTRINSICS.XI CONTAINS n CARDS
    <n sorted patches>
##PATCH NUMBER xx FOR INTRINSICS.XI CONTAINS n CARDS
    <n sorted patches>
? END

```

The \$INTRINSIC Option is required when compiling the symbolic file for the INTRINSICS of the B 5500 Programming System. When compiling these INTRINSICS, the ESPOL file DISK is equated to INT/DISK.

NOTE

When the INTRINSICS are compiled, it is necessary to execute the FILL/PRT Program which builds the MCP/PRT Disk File used by DUMP/ANALYZE. If the INTRINSICS are compiled in the manner shown, FILL/PRT should be executed as follows:

```

? EXECUTE FILL/PRT
? FILE MCP = 0000000/MCP DISK
? FILE INT = 0000000/INT DISK
? END

```

An example of a basic change deck format for the ESPOL Compiler is as follows:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR ESPOL.XI CONFLICTS
$*COMPILE ESPOL/DISC ALGOL LIBRARY
$*ALGOL PROCESS = 60; ALGOL IO = 60
$*ALGOL FILE TAPE = SYMBOL/ESPOLSY
$*DATA CARD
$-CARDS NEEDED FOR ESPOL COMPILATION
$ TAPE LIST PRT CHECK

```

```

##PATCH NUMBER 1 FOR ESPOL.XI CONTAINS n CARDS
    <n sorted patches>
##PATCH NUMBER xx FOR ESPOL CONTAINS n CARDS
    <n sorted patches>
? END

```

Following is an example of a basic change deck format for the ALGOL Compiler:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR ALGOL.XI CONFLICTS
$*COMPILE ALGOL/DISC ALGOL LIBRARY
$*ALGOL PROCESS = 60; ALGOL IO = 60
$*ALGOL FILE TAPE = SYMBOL/ALGOLSY
$*FILE LINE = LINE BACK UP DISK
$*FILE NEWTAPE = "OCRDIMG" TAPE
$*FILE PNCH = PNCH PUNCH
$*DATA CARD
$-CARDS NEEDED FOR ALGOL COMPILATION
$ TAPE LIST PRT CHECK
##PATCH NUMBER 1 FOR ALGOL.XI CONTAINS n CARDS
    <n sorted patches>
##PATCH NUMBER xx FOR ALGOL.XI CONTAINS n CARDS
? END

```

NOTE

The same setup is used to compile XALGOL (SYMBOL/XALGLSY).

An example for the COBOL Compiler is as follows:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR COBOL.XI CONFLICTS
$*COMPILE COBOL/DISC ALGOL LIBRARY
$*ALGOL STACK = 750

```

```

$*ALGOL PROCESS = 60; ALGOL IO = 60
$*ALGOL FILE TAPE = SYMBOL/COBOLSY
$*FILE NEWTAPE = SOLT TAPE
$*FILE LINE = LINE BACK UP DISK
$*DATA CARD
$-CARDS NEEDED FOR COBOL COMPILATION
$ TAPE LIST PRT CHECK
$#PATCH NUMBER 1 FOR COBOL.XI CONTAINS n CARDS
    <n sorted patches>
$#PATCH NUMBER xx FOR COBOL.XI CONTAINS n CARDS
    <n sorted patches>
? END

```

Following is an example for the FORTRAN Compiler:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR FORTRAN.XI CONFLICTS
$*COMPILE FORTRAN/DISC ALGOL LIBRARY
$*ALGOL PROCESS = 60; ALGOL IO = 60
$*ALGOL FILE TAPE = SYMBOL/FORTSY
$*FILE LINE = LINE BACK UP DISK
$*FILE NEWTAPE = FORTSYN/TAPE
$*DATA CARD
$-CARDS NEEDED FOR FORTRAN COMPILATION
$ TAPE LIST PRT CHECK
$#PATCH NUMBER 1 FOR FORTRAN.XI CONTAINS n CARDS
    <n sorted patches>
$#PATCH NUMBER xx FOR FORTRAN.XI CONTAINS n CARDS
    <n sorted patches>
? END

```

An example for the BASIC Compiler is as follows:

```

? EXECUTE PATCH/MERGE
? DATA CARD
$.xx PATCHES FOR BASIC.XI CONFLICTS

```



```

$*COMPILE BASIC/DISK ALGOL LIBRARY
$*ALGOL PROCESS = 60; ALGOL IO = 60
$*ALGOL FILE TAPE = SYMBOL/BASICSY TAPE
$*DATA CARD
$-CARDS NEEDED FOR BASIC COMPILATION
$ TAPE LIST PRT CHECK
$#PATCH NUMBER 1 FOR BASIC.XI CONTAINS n CARDS
    <n sorted patches>
$#PATCH NUMBER xx FOR BASIC.XI CONTAINS n CARDS
    <n sorted patches>
? END

```

Following is an example for Card Load Select Programs:

a. COLD START.

```

? EXECUTE ESPOL/DISK .COMPILE MKXI COLD START
? FILE LINE = LINE BACK UP DISK
? FILE TAPE = SYMBOL/COOLSY
? DATA CARD
$ SET COOL = FALSE
$ TAPE LIST PRT DECK

```

99999999

? END

b. COOL START.

```

? EXECUTE ESPOL/DISK .COMPILE MKXI COOL START
? FILE TAPE = SYMBOL/COOLSY
? FILE LINE = LINE BACK UP DISK
? DATA CARD
$ SET COOL = TRUE
$ TAPE LIST PRT DECK

```

99999999

? END

c. DUMP.

```

? EXECUTE ESPOL/DISK .COMPILE DUMP CORE TO TAPE
? FILE TAPE = SYMBOL/DUMPTAP

```

? FILE LINE = LINE BACK UP DISK

? DATA CARD

\$ TAPE LIST PRT DECK

99999999

? END

SYSTEM TAPE.

The SYSTEM Tape is a library tape which contains object programs (routines) which have been compiled from the SYMBOL Tape(s). (Refer to table 3-1.) The purpose of the SYSTEM Tape is to have all routines, compilers, and operating systems that are required for loading and maintaining the system on one tape in an effort to eliminate wasted time and trouble. The SYSTEM Tape has the identification of SYSTEM/FILE000 and is created by use of the Dump Control Card which specifies the <multifile ID> // <file ID> of object programs which have been compiled from the SYMBOL Tape.

The position of a file on the SYSTEM Tape should be determined by the use of that file (i.e., the number of times a file may be expected to be loaded or reloaded from the SYSTEM Tape); it is recommended that the MCP be the first file, the INTRINSICS second, the compilers next, followed by other routines and user programs.

Because of the importance of the SYSTEM Tape, each installation should make a backup copy of it. This can be done with a Dump Control Card.

CARD LOAD SELECT PROGRAMS.

A Card Load Select Program is an object (machine language) routine created by the ESPOL Compiler; it is loaded into the system through Card Reader A with the console CARD LOAD SELECT switch/indicator in the on position (lit).

The Cold Start, Tape to Disk, and Cool Start Card Decks may not be used if an MCP is running on any of the systems attached to File Protect Memory. Disk to Disk and Kernel Card Decks, however, may be run while an MCP is operating on one or more systems.

The Card Load Select Programs are listed below and described in the following paragraphs.

- a. ESPOL Loader.
- b. Halt/Load.
- c. Cold Start.
- d. Cool Start.
- e. Tape to Disk MCP Loader.
- f. Disk to Disk MCP Loader.
- g. Halt/Load Kernel.
- h. Core to Tape Dump.

ESPOL LOADER PROGRAM. The ESPOL Loader Program is a 1-card Card Load Select Program which has the primary function of properly loading the object card program that follows this card.

HALT/LOAD PROGRAM. The Halt/Load Program is a 1-card Card Load Select Program which has the primary function of calling on object routines to load the MCP from disk into core memory and initiate running.

COLD START PROGRAM. The Cold Start Program is a Card Load Select Program which has the primary function of:

- a. Creating an initial Disk Directory.
- b. Building the skeleton of, and placing the number of the systems and the address of DIRECTORYTOP into, segment zero while "zeroing out" the rest of the segment.
- c. Setting initial operating conditions.

The Cold Start Program must be run before the MCP is used for the first time since the MCP expects to find the Disk Directory.

NOTE

The Cold Start Program is generated with \$ COOL set to FALSE in the Cold Start Program Basic Change Deck. (Refer to the Cold Start example on page 3-13.)

COOL START PROGRAM. The Cool Start Program is a Card Load Select Program which has the primary function of searching the Disk Directory and removing each file that has the following characteristics:

- a. The first character of the multiple-file-identification or file-identification is not zero.
- b. Any word in the file header has the flag bit ON.
- c. The maximum number of declared areas is greater than 20.
- d. The disk address of an area is less than DISKTOP or greater than that possible, based on the number of disk electronics units declared.
- e. A disk address is present for an area that is outside the maximum number of declared areas.

The names of the removed files are printed on the SPO if the portion of the Disk Directory entry containing the file name is intact. The removed files may be reloaded from tape after the system has undergone a Halt/Load.

The Cool Start Program also checks to see if it might remove the MCP. If so, a message is printed on the SPO to indicate that a Tape to Disk should be performed to reload the MCP. Cool Start also removes the intrinsics for the system on which Cool Start is being run. When Cool Start finishes checking the Disk Directory, the operator may run Tape to Disk to reload the MCP.

NOTE

The Cool Start Program is generated by setting \$ COOL to TRUE in the Cool Start Program Basic Change Deck. (Refer to Cool Start example on page 3-13.)

TAPE TO DISK MCP LOADER PROGRAM. The Tape to Disk MCP Loader Program is a Card Load Select Program which has the primary function of:

- a. Loading an MCP from tape to disk.
- b. Updating the Disk Directory Header to actual MCP size.
- c. Updating segment zero to show the actual name and address of the MCP that is being loaded.

DISK TO DISK MCP LOADER PROGRAM. The Disk to Disk MCP Loader is a Card Load Select Program which has the primary function of updating segment zero to show the name and address of an MCP which is on disk and on which the system is to be run at this time.

HALT/LOAD KERNEL PROGRAM. Halt/Load Kernel is a Card Load Select Program which is used for clearing File Protect Memory and/or loading the MCP from disk into core memory. Halt/Load Kernel may be run independently of MCP Loader Decks and/or System Loader Decks. Following is the skeleton deck setup for the independent running of Halt/Load Kernel:

- a. ESPOL Loader Card Program.
- b. Halt/Load Kernel Program.
- c. Halt/Load Card Program.

CORE TO TAPE DUMP PROGRAM. Core to Tape Dump is a Card Load Select Program used for debugging analysis. This program is run independently of all other decks. It has the following deck setup structure:

- a. Two-card ESPOL Loader.
- b. Core to Tape Dump Program.

After Core to Tape Dump is executed, it inquires as to which tape unit the contents of memory is to be written by typing out the WHICH TAPE? message on the SPO. One of the valid replies to WHICH TAPE? might be MTA.

MCP LOADER DECKS.

MCP Loader Decks are combinations of Card Load Select Programs which function to load an MCP and/or update segment zero in the Disk Directory.

TAPE TO DISK MCP LOADER DECK. Following is the skeleton setup of the Tape to Disk MCP Loader Deck:

- a. ESPOL Loader Card (program).
- b. Tape to Disk Loader (program).
- c. Halt/Load Kernel (optional program).
- d. Tape to Disk Parameter (specifier) Cards.
- e. Halt/Load Card (program).

The Tape to Disk MCP Loader Deck loads an MCP file, MCP/DISK, from a library tape, SYSTEM/FILE000, to an existing disk area. The disk area to which the MCP file is loaded must have the identical file name of the MCP file being loaded and should be declared in the Cold Start Parameter Deck. Tape to Disk Parameter Cards may be supplied to alter the sought \langle multifile ID \rangle/\langle file ID \rangle of an MCP file and/or the \langle multifile ID \rangle of a library tape.

The Tape to Disk Parameter Cards that may be used are:

- a. FILE = \langle multifile ID \rangle/\langle file ID \rangle
The Loader loads the \langle multifile ID \rangle/\langle file ID \rangle file from the tape to disk.
- b. TAPE = \langle file ID \rangle
The name of the library tape from which the MCP is to be loaded is \langle file ID \rangle .

If the Kernel Deck is present, i.e., if the Kernel Deck is located after the Tape to Disk Loader and before any (optional) Tape to Disk Parameter Cards, it is loaded.

DISK TO DISK MCP LOADER DECK. Following is the skeleton setup of the Disk to Disk MCP Loader Card Deck:

- a. ESPOL Loader Card (program).
- b. Disk to Disk MCP Loader (program).
- c. Halt/Load Kernel (optional program).
- d. Disk to Disk Parameter (specifier) Card.

The Disk to Disk MCP Loader assumes the Disk Directory already exists; i.e., the Disk to Disk Loader is not a replacement for the Tape to Disk Loader. The Disk to Disk MCP Loader takes the MCP name from the parameter card after the Kernel and updates segment zero with the name and address of the MCP. The parameter card may be coded in free form (i.e., MCP/DISK or MCP DISK). A Disk to Disk must be performed on each additional system after a Cold Start. The Loader does not need the Kernel if a valid Kernel is already present on disk.

If the file name specified in the parameter card is not in the Disk Directory, a message to this effect is printed and another card is read. If the specified file is in the Directory, the SPO skips one line and the file is loaded. The message <file specifier> LOADED is typed, and a Halt/Load sequence is initiated.

SYSTEM LOADER DECKS.

System Loader Decks are combinations of Card Load Select Programs and MCP Loader Decks which function to initialize or re-instate the Disk Directory and MCP. System Loader Decks are classified as the following:

- a. Cold Start Deck.
- b. Cool Start Deck.

These card decks are described in the following paragraphs.

COLD START DECK. Following is the skeleton setup of the Cold Start Deck:

- a. ESPOL Loader Card Program.
- b. Cold Start Program.
- c. Cold Start Parameter Cards.
- d. Tape to Disk MCP Loader Deck.
 - 1) ESPOL Loader Card Program.
 - 2) Tape to Disk Loader Program.
 - 3) Halt/Load Kernel Program.

- 4) Tape to Disk Parameter Cards (optional).
 - 5) Halt/Load Card Program.
- e. Control Cards for system loading.

The Cold Start Program constructs the initial Disk Directory and initializes the current date word and option codes. It builds the skeleton of segment zero and puts the number of systems and the address of DIRECTORYTOP into segment zero while zeroing out the rest of that segment.

The Cold Start Parameter Cards have a free-field format. They are described in the following paragraphs.

NOTE

The last Cold Start Parameter Card must be a Stop Card, i.e., a card containing the word STOP. Three or more parameter cards must immediately precede the Stop Card.

DRCTRYTP Card. *See also FENCE card, page 3-22.*

The DRCTRYTP (Directory Top) Card provides an integer which specifies the absolute disk address of the segment known as DIRECTORYTOP. A DRCTRYTP Card must appear in the Cold Start Deck. It must be the first parameter card in the deck.

The DRCTRYTP Card must have the following information:

DRCTRYTP <integer> *See MCP Reference Manual, page 2-2.*

Example

DRCTRYTP 1199

DIRECT Card. *See also page 5-41*

The DIRECT Card provides an integer which specifies the address of the highest addressed disk segment which should be used for the Disk Directory. A DIRECT Card must appear in the Cold Start Deck.

When determining the figure to be specified as the upper boundary, it should be realized that every 15 files on the disk use 16 segments in the Directory.

notes: The Disk directory once overflowed with DRCRTTP = 1203 DIRECT = 2503 (1296 segments) even though only 899 files were detected by DISKDIR/UTILITY (this is a bug)

The DIRECT Card must have the following information:

DIRECT <integer>

Example

DIRECT 2500

ESU Card.

The ESU (electronics storage unit) Card supplies an integer which indicates the number of electronics units connected to the system. If there are no electronics units on the system which are unique to DKB (i.e., Disk File Control Unit 2), this card should contain an integer which reflects the total number of electronics units on the system. (This is generally the case when an exchange is present.) If there are electronics units which are unique to DKB, the integer on this card must have a value which expresses the sum of the number of electronics units on DKA (i.e., Disk File Control Unit 1) plus 100 times the number of electronics units on DKB. For example, if a system has three electronics units on DKA and two on DKB, the integer on the ESU Card should be 203. An ESU Card must appear in the Cold Start Deck.

The ESU Card must contain the following information:

ESU <integer>

Examples

ESU 1

ESU 202

SYSTEMS Card.

To specify the number of systems, the SYSTEMS Card is required in the Cold Start Deck.

simultaneously running independent MCP

The SYSTEMS Card must contain the following information:

SYSTEMS = <physical number of systems>

Examples

SYSTEMS = 1

SYSTEMS = 4

FENCE Card. See also DIRECTORYTOP card, page 3-20

The FENCE Card initializes DIRECTORYTOP [19] which is used by the Time Sharing MCP only. The purpose of the "fence" is to separate memory into two areas:

- a. Overlayable - that area occupied by the MCP, MCP Tables, and Run jobs. *Time Sharing*
- b. Swapping - that area occupied by Execute jobs. *Time Sharing*

The format of the FENCE Card is:

FENCE = <integer>

NOTE

Any <integer> less than 8192 is changed to 8192 by the system. Any <integer> greater than 28582 is changed to 28582. Any <integer> is changed to the nearest number that is a multiple of 1024.

DATE Card.

The DATE Card is optional and should precede all FILE Cards in the Cold Start Deck, if any. A date supplied in a DT keyboard message, entered subsequently to the use of the DATE Card, supersedes the information in the DATE Card.

The DATE Card provides three integers separated by the character /. The first integer specifies the two digits of the month, the second integer specifies the day, and the third specifies the last two

digits of the year. This card causes the date-word on the disk to be set to the date specified. (This date-word contains the current date used, e.g., in tape labels.)

The DATE Card must contain the following information:

DATE <integer> / <integer> / <integer>

Example

DATE 12/29/64

FILE Card Group.

The function of a FILE Card Group is to define a user file which is to be listed in the Disk Directory. This method of defining a file, as opposed to defining a file through use of a file declaration in a program, allows an installation to explicitly assign specific disk addresses for files.

A FILE Card Group consists of a FILE Card and one or more File Address Cards. There may be as many FILE Card Groups in the Cold Start Deck as desired.

The FILE Card contains the word FILE which identifies the FILE Card Group. It supplies the following information in the order listed. The information is separated by commas. (Refer to the FILE Card examples in section 4.)

- a. Data file specifier which provides the data file name to be listed in the Disk Directory.
- b. <integer> (x) <integer> construct, where the first integer specifies the number of areas on disk to be used by the file, and the second integer specifies the number of 30-word disk segments in each area. (The symbol (x) in this construct is the multiply character.)
- c. Integer which specifies the purge factor for the file (i.e., the number of days past the date of last access that the file is to be retained on disk).

In summary, a FILE Card must contain the following information:

FILE <data file specifier> , <integer> (x) <integer> , <integer>

Example

FILE PREFIX/NAME , 2 (x) 1000, 30

A File Address Card contains a single integer which, if different from zero, specifies the absolute disk address of the first word in an area to be used by the file whose name is supplied by the preceding FILE Card. There must be one File Address Card for each area specified for the file. The first File Address Card in a FILE Card Group provides the beginning address of the first area to be used by the file; the second File Address Card provides the beginning address of the second area to be used; and so on. A zero integer for an address denotes that the MCP is to assign the address for that area.

As noted above, a File Address Card must contain the following information:

<integer>

Examples

2000

0

Three examples of a FILE Card Group are as follows:

a. FILE SYSTEM/LOG, 1 (x) 1000, 2
0

b. FILE SAVES/AREA, 3 (x) 9000, 15
2500
3400
0

c. FILE B 280/RESERVE, 2 (x) 1500, 365
4300
8000

It should be noted that if log information for the system is to be recorded, a file with the file identification prefix SYSTEM and the file identification LOG must be defined on the disk. Also, this file must be limited to one area on the disk. Consequently, if system log information is to be retained, a FILE Card Group (such as the first example above) should appear in the Cold Start Deck.

The following list contains examples of FILE Cards which are required for the given circumstances:

a. General.

- 1) FILE ^{DIRCTRY}~~DIRCTRY~~/DISK, 1 (x) <size>, <save factor>
 <address = DIRECTRYTP+4>
- 2) FILE MCP/DISK, 1 (x) ¹⁵⁹⁰~~1170~~, 999
 <address = <DIRCTRY/DISK address>+<~~size~~>+1>
- size = DIRECT - DIRCTRYTP - 4
-

b. Dump Analyzer.

- FILE DMPAREA/DISK, 1 (x) 100, 999
 <address = <MCP/DISK address>+<¹⁵⁹⁰~~1170~~>+1>
-] address has to agree with address punched in 2-card prefix to memdump deck

c. Systems Log.

- FILE SYSTEM/LOG, 1 (x) <size>, <save factor>
 <address = <DMPAREA/DISK address>+<¹⁰⁰~~size~~>+1>

d. Statistics Pseudo Logs.

1) Macro-Temporary.

FILE SYSTEM<system mnemonic>/STATS, 1 (x) <size>, <save factor>
 <address = <SYSTEM/LOG address>+<size>+1>

2) Macro-Permanent

FILE SYSTAT<system mnemonic>/DISK, 1 (x) <size>, <save factor>

<address = <SYSTEM<system mnemonic>/STATS address>+
<size>+1>

3) Micro-Permanent.

FILE STLOG<system mnemonic>/STATS,1 (x) <size>,<save
factor>

<address = <SYSTAT<system mnemonic>/DISK address>+
<size>+1>

OPTION Cards.

If the option cards described in the following paragraphs are not used, the corresponding options are automatically set to OFF.

The USE DRA or OPTN 47 Card specifies that a system is equipped with a drum memory unit designated DRUM A. Consequently, this option can be used only when the system is so equipped. When this option is specified, the MCP uses DRUM A for overlay storage in preference to disk. When the USE DRA Option is specified, a Loader which loads the MCP from disk to core at Halt/Load time is placed on the drum so that a Halt/Load from the drum can be used to initiate the operation of the system. However, in this case only, a Halt/Load Card must physically be the last card in the Cold Start Deck, thereby making the STOP Card the next-to-the-last card (see below). If the USE DRA Option is not used, a CARD LOAD SELECT, HALT, and LOAD must be performed to initiate the system. This is done by using the Disk Halt/Load Card which is described later in this section.

The USE DRA Card must contain either USE DRA or OPTN 47.

Examples

USE DRA

OPTN 47

The USE DRB or OPTN 46 Card specifies that a system is equipped with a drum memory unit designated DRUM B. Consequently, this option can only be used when the system is so equipped. When this option is

specified, the MCP uses DRUM B for data overlay storage, and, when available, drum memory is used in preference to disk memory.

The USE DRB Card must contain either USE DRB or OPTN 46.

Examples

USE DRB
OPTN 46

The TYPE BOJ or OPTN 45 Card specifies that a BOJ message is to be typed each time the MCP initiates a compiler or an object program.

The TYPE BOJ Card must contain either TYPE BOJ or OPTN 45.

Examples

TYPE BOJ
OPTN 45

The TYPE EOJ or OPTN 44 Card specifies that an EOJ message is to be typed by the MCP when a compiler or an object program has come to a normal completion.

The TYPE EOJ Card must contain either TYPE EOJ or OPTN 44.

Examples

TYPE EOJ
OPTN 44

The TYPE OPEN or OPTN 43 Card specifies that a message is to be typed by the MCP whenever an object program opens a file other than a disk file.

The TYPE OPEN Card must contain either TYPE OPEN or OPTN 43.

Examples

TYPE OPEN
OPTN 43

The USE TERMINATE or OPTN 42 Card specifies that the Terminate Procedure of the MCP is to be called if the MCP must discontinue processing of a program because of an error condition.

Since it is the function of the Terminate Procedure to clear the system of all information pertaining to a discontinued program, the USE TERMINATE Option generally should always be specified. However, if an error condition should occur where it is necessary to obtain a memory dump that reflects core conditions at error time, the USE TERMINATE Option should not be used.

The USE TERMINATE Card must contain either USE TERMINATE or OPTN 42.

Examples

```
USE TERMINATE
OPTN 42
```

- The TYPE DATE or OPTN 41 Card specifies that #DT PLEASE is to be typed by the MCP at Halt/Load time and that the system operator must enter a DT keyboard input message before processing can commence.

The TYPE DATE Card must contain either TYPE DATE or OPTN 41.

Examples

```
TYPE DATE
OPTN 41
```

The TYPE TIME or OPTN 40 Card specifies that TR PLEASE is to be typed by the MCP at Halt/Load time and that the system operator must enter a TR keyboard input message before processing can commence.

The TYPE TIME Card must contain either TYPE TIME or OPTN 40.

Examples

```
TYPE TIME
OPTN 40
```


OPTN 39, ONEBREAK, is no longer used and should be reset. ■

The USE AUTOPRNT or OPTN 38 Card specifies that printer backup tapes (not including those created previously to the latest Halt/Load) are to be automatically printed when a backup tape and a line printer are not in use at the same time. If the option is not specified, printer backup tapes are printed only if the system operator enters a PB keyboard input message.

The USE AUTOPRNT Card must contain either USE AUTOPRNT or OPTN 38.

Examples

```
USE AUTOPRNT
OPTN 38
```

The USE CLEARWRS or OPTN 37 Card specifies that the MCP is to attempt to keep remote data communications stations (which have SPO capabilities) from remaining in a Write-Ready Condition if object program output is not available when the condition occurs. (The Write-Ready Condition occurs when an output message which does not end in a group mark is sent to a data communications station. When a data communications station is Write-Ready, input cannot be received.) If a Write-Ready Condition occurs when no more output is queued for a station, if the station has SPO capabilities, and if the USE CLEARWRS Card is used, the MCP sends a group mark to the station, thus clearing the Write-Ready.

NOTE

This option is applicable to data communications systems only.

The USE CLEARWRS Card must contain either USE CLEARWRS or OPTN 37.

Examples

```
USE CLEARWRS
OPTN 37
```

The TYPE DISCONDC or OPTN 36 Card specifies that the MCP is to write a disconnect code on any data communications station which is not logged in when the station is disconnected from a program. This option should not be set if any data communications equipment on the system is connected to any telephone company equipment not wired to handle disconnect codes.

NOTE

This option is applicable to data communications systems only.

The TYPE DISCONDC Card must contain either TYPE DISCONDC or OPTN 36.

Examples

```
TYPE DISCONDC
OPTN 36
```

The TYPE CMPLFILE or OPTN 35 Card specifies that file-open and file-close messages are to be typed for compiler files according to the respective settings of the TYPE OPEN and TYPE CLOSE options. If this option is not specified, messages are not typed because of the opening and/or closing of files used by compilers.

The TYPE CMPLFILE Card must contain either TYPE CMPLFILE or OPTN 35.

Examples

```
TYPE CMPLFILE
OPTN 35
```

The TYPE CLOSE or OPTN 34 Card specifies that a message is to be typed by the MCP whenever an object program closes a file other than a disk file.

The TYPE CLOSE Card must contain either TYPE CLOSE or OPTN 34.

Examples

```
TYPE CLOSE
OPTN 34
```

The TYPE ERRORMSG or OPTN 33 Card specifies that object-time error messages are to be typed by the MCP if errors are encountered during the running of an object program and if programmatic recovery is used in the program.

The TYPE ERRORMSG Card must contain either TYPE ERRORMSG or OPTN 33.

Examples

```
TYPE ERRORMSG
OPTN 33
```

The TYPE RET or OPTN 32 Card specifies that magnetic-tape retention messages are to be typed by the MCP.

The TYPE RET Card must contain either TYPE RET or OPTN 32.

Examples

```
TYPE RET
OPTN 32
```

The TYPE LIBMSG or OPTN 31 Card specifies that the following messages are to be typed by the MCP when appropriate:

```
<program ID> LOADED <unit mnemonic>
<program ID> DUMPED <unit mnemonic>
<program ID> REMOVED
<program ID> CHANGED TO <program ID>
```

The TYPE LIBMSG Card must contain either TYPE LIBMSG or OPTN 31.

Examples

```
TYPE LIBMSG
OPTN 31
```

The TYPE SCHEDMSG or OPTN 30 Card specifies that a message is to be typed by the MCP whenever a job is placed in the schedule.

The TYPE SCHEDMSG Card must contain either TYPE SCHEDMSG or OPTN 30.

Examples

TYPE SCHEDMSG
OPTN 30

The TYPE SECMSG or OPTN 29 Card specifies that File Security maintenance messages are to be typed by the MCP.

NOTE

This option is applicable to data communications systems only.

The TYPE SECMSG Card must contain either TYPE SECMSG or OPTN 29.

Examples

TYPE SECMSG
OPTN 29

The USE DSKTOG or OPTN 28 Card specifies that any object program attempting input or output at any absolute disk address below the user disk area is to be discontinued by the MCP, and an INVALID PRL message is to be typed.

NOTE

This option is applicable to data communications systems only.

The USE DSKTOG Card must contain either USE DSKTOG or OPTN 28.

Examples

USE DSKTOG
OPTN 28

The USE RELTOG or OPTN 27 Card specifies that all object programs running under File Security which attempt to perform an ALGOL RELEASE Statement referencing disk are to be automatically discontinued by the MCP.

NOTE

This option is applicable to data communications systems only.

The USE RELTOG Card must contain either USE RELTOG or OPTN 27.

Examples

```
USE RELTOG
OPTN 27
```

The TYPE PBDREL or OPTN 26 Card specifies that printer backup disk-release messages are to be typed by the MCP when OPTN 38, USE AUTOPRNT, is not set.

The TYPE PBDREL Card must contain either TYPE PBDREL or OPTN 26.

Examples

```
TYPE PBDREL
OPTN 26
```

The USE CHECK or OPTN 25 Card specifies that the MCP is to check all memory links for validity whenever allocation or deallocation of memory is performed. If an invalid link is found, the message INVALID LINK is printed on the SPO. A Halt/Load must then be performed. This option (refer to section 4) is available only when the MCP is compiled with:

```
$ DEBUGGING=TRUE
```

If this option is not set (reset) or if the MCP is compiled with DEBUGGING=FALSE, the MCP only checks adjoining links when allocating or deallocating memory.

The USE CHECK Card must contain either USE CHECK or OPTN 25.

Examples

```
USE CHECK
OPTN 25
```

The TYPE DISKMSG or OPTN 24 Card specifies that disk retry messages are to be typed by the MCP even though retrying is successful. If this option is not set (reset), the retry message is provided only if the retry is not successful.

The TYPE DISKMSG Card must contain either TYPE DISKMSG or OPTN 24.

Examples

TYPE DISKMSG

OPTN 24

The TYPE DISKLOG or OPTN 23 and TYPE LIBERR or OPTN 22 Options are used only in the Time Sharing System MCP.

The USE PBDONLY or OPTN 21 Card specifies that all output to the line printers (except from printer backup) is forced to printer backup disk.

The USE PBDONLY Card must contain either USE PBDONLY or OPTN 21.

Examples

USE PBDONLY

OPTN 21

The USE SAVEPBT or OPTN 20 Card specifies that all printer backup tapes are to be rewound and locked by the MCP when released by the creating program.

The USE SAVEPBT Card must contain either USE SAVEPBT or OPTN 20.

Examples

USE SAVEPBT

OPTN 20

The TYPE RSMMSG or OPTN 19 Card specifies that a message is to be typed by the MCP whenever the "access flag" on a disk file is set or reset by a control card.

The TYPE RSMSG Card must contain either TYPE RSMSG or OPTN 19.

Examples

TYPE RSMSG
OPTN 19

The USE AUTOUNLD or OPTN 18 Card is used to specify system action when a NO USER DISK message occurs. If the option is set when a NO USER DISK occurs, the operator must type in <mix number>WY; the MCP then initiates an UNLOAD EXPIRED TO XP <Julian date>=/=. The control card is built at sequences 06365230-06365250 in the MCP and may be changed if it is desired to dump specific files, and so forth. The AUTOUNLD Option is reset following the UNLOAD Operation.

Examples

USE AUTOUNLD
OPTN 18

STOP Card.

The STOP Card must physically be the last card of the Cold Start Deck and must contain STOP; at least three parameter cards must precede the STOP Card.

Example

STOP

COOL START DECK. Following is the skeleton setup of the Cool Start Deck:

- a. ESPOL Loader Card Program.
- b. Cool Start Program.
- c. Cold Start Parameter Cards (except File Cards).
- d. MCP Loader Deck (optional).
 - 1) Tape to Disk MCP Loader Deck (tape option).
 - a) ESPOL Loader Card Program.
 - b) Tape to Disk MCP Loader Program.

- c) Halt/Load Kernel Program.
- d) Tape to Disk Parameter Cards.
- 2) Disk to Disk MCP Loader Deck (disk option).
 - a) ESPOL Loader Card Program.
 - b) Disk to Disk MCP Loader Program.
 - c) Halt/Load Kernel Program.
 - d) Disk to Disk Parameter Card.
- e. Halt/Load Card Program.

The Cool Start Program should be run whenever the Disk Directory appears to contain incorrect entries. Use of the Cool Start Program, in most cases, eliminates the necessity of performing a Cold Start when consistent system problems are occurring. The following conditions are good indications that the Directory contains incorrect entries:

- a. The system "hangs up" during a Halt/Load.
- b. The system attempts to address an electronics unit which is not on-line.
- c. The system attempts to address a disk module which is not on-line.
- d. The system hangs up during disk parity retry operations.
- e. Any program accessing the Directory is terminated with a -FLAG BIT error message.
- f. Several system hang-ups occur in a short period of time.

The Cool Start Parameter Deck is set up identically to the Cold Start Deck with one exception: The File Cards must not be included.

CONTROL CARDS FOR SYSTEM LOADING.

At the end of a Cold Start Deck, control card parameters may be entered to load the System Files.

The LOAD Cards which cause programs to be read from the System Tape and entered as library files on the user disk are identical in

construct to the LOAD Cards described in section 4 of this document. For example, a LOAD Card to load the ALGOL Compiler onto disk could appear as follows:

```
? LOAD FROM SYSTEM ALGOL/DISK
```

DISK HALT/LOAD CARD.

The Disk Halt/Load Card is a binary card containing code that causes the MCP to be initiated. This card must be used in conjunction with a CARD LOAD SELECT, HALT, and LOAD Operation to initiate the MCP. The only exceptions to this occur when the system is equipped with a drum memory unit and the DRA Option is specified, or when the system has the Disk Halt/Load hardware feature installed. (Refer to the Halt/Load Program description on page 3-15.)

CONTROL CARDS USED TO LOAD COMPILERS ONTO DISK.

The control cards used in the System Loader to load compilers onto disk are:

- a. A LOAD Card specifying the library tape name SYSTEM and the names of the compilers to be loaded.
- b. An END Card.

The following example shows a possible choice of such control cards:

```
? LOAD FROM SYSTEM ALGOL/DISK, COBOL/DISK  
? END
```

SYSTEMS PROCEDURES.

The following procedures are supplied for loading and maintaining the system:

- a. System start-up.
- b. Loading the system from the System Tape(s).
- c. Program scheduling.
- d. Selection Algorithm.
- e. Multiprocessing Factor.

These procedures are described in the following paragraphs.

SYSTEM START-UP PROCEDURE.

Before beginning any operation, the system must be turned on by pressing the POWER ON switch on the operator console. All other hardware units which are to be used must be readied as described in section 2.

LOADING THE SYSTEM FROM THE SYSTEM TAPE.

The entire software package (MCP, compilers, system programs, etc.) is contained on the System Tape. Any or all the above items can be loaded onto disk by means of the System Loader. This loading can be accomplished all at one time or at different times. Once the desired items have been placed on disk, they remain there until a REMOVE Operation is called for.

To load the system with all items on the System Tape at the same time, the operator must perform the following steps:

- a. Press the CARD LOAD SELECT and HALT switches on the operator console. (These switches light when in the conditions shown on the switches.)
- b. Place the System Tape, without a write ring, on a tape unit.
- c. Press LOCAL, then LOAD on the tape unit.
- d. When the tape reels stop moving, press REMOTE on the tape unit.
- e. Place the System Loader Card Deck in the card reader.
- f. Press RESET, then START on the card reader.
- g. Press LOAD on the operator console.

Since the MCP Loader and the Cold Start are independent routines, they may be loaded onto disk separately or together by placing the respective card decks in the card reader and performing a CARD LOAD SELECT, HALT, and LOAD Operation. Also, the Disk Halt/Load Card can be used

anytime the MCP and the Disk Directory are on disk. The control cards used to load the compilers may be used any time the MCP is in operation. Therefore, the MCP Loader, the Cold Start, the Disk Halt/Load, or the Software Load Cards may be used separately. For example, if a compiler that has not been loaded is required on disk, a Load Card alone can be used to place the compiler on disk. If a modified MCP is to replace the MCP currently on disk, the MCP Loader alone can be initiated with a CARD LOAD SELECT, HALT, and LOAD Operation.

There are messages which are typed out while loading the system. They are as follows:

<u>Message</u>	<u>Description</u>
MCP FILE LOADED	The MCP Loader has successfully loaded the MCP onto disk.
DIRECTORY BUILT	Cold Start has constructed a Disk Directory.
-H/L-	The MCP has assumed control of the system.
INCORRECT CARD	An erroneous card is read during the operation of the Cold Start. This situation can be remedied by placing the correct card in the card reader, pressing RESET, then START on the card reader.
<unit mnemonic> ERROR	Self explanatory.

PROGRAM SCHEDULING INFORMATION.

After the desired information has been loaded onto disk, the MCP is ready to begin processing. The MCP is "told" what is to be processed by means of control information. This information, made available to the MCP via punched cards, is placed in an available card reader by the operator.

The MCP scans the available input/output units and reads a record from each input file. During this scanning operation, the MCP recognizes the card it reads as containing the needed control information. The card is then analyzed, and the indicated operation is performed.

THE SELECTION ALGORITHM.

■ The Selection Algorithm in the MCP attempts to prevent too many programs from being executed at the same time; otherwise, the number of programs which are multiprocessed together would be limited only by MIXMAX. Presented with too many jobs at the same time, the system might load all the jobs into core memory with the following possible results:

- a. The system might run out of core memory, thereby requiring a Halt/Load.
- b. The system might become overlay-bound, with the total elapsed time for the multiprocessed mix greater than that required to run the jobs serially.

The Selection Algorithm runs only as many programs that fit together in core. Estimates of core requirements for programs are constructed by the ALGOL, FORTRAN, and COBOL Compilers. The MCP determines how much core memory is available for the processing of object programs. As each program is started, its core requirements are added to the total. A program is not started if the sum of its core requirements plus the sum of the core requirements for jobs already running exceeds the total amount of core storage available for the processing of object programs.

The Selection Algorithm contains several variants which make it more useful. These are as follows:

- a. Any program is loaded immediately, regardless of its core requirement, if there are no other jobs running.

- b. The core estimates created by the compilers are reasonably accurate. It should be understood, however, that it is logically impossible for a compiler to determine exactly how much core space a program needs in order to run efficiently. (The compiler could easily determine a maximum core requirement, but program logic determines which program segments, arrays, and files should coexist in core storage for efficient operation.) A program-parameter card, called a CORE Card, is provided to allow the programmer or operator to override the core requirement provided by the compiler.
- c. Programs which have been presented to the MCP to be run, but which have not been started because of the lack of core space, are said to be in the "schedule."
- d. The operator may choose to cause a program which is in the schedule to be loaded in spite of the fact that the MCP does not think the program will run efficiently with the jobs already in the mix. (Refer to XS input message in appendix C.)
- e. The operator may choose to terminate a program which is still in the schedule. (Refer to ES input message in appendix C.)

THE MULTIPROCESSING FACTOR.

At Halt/Load time, the MCP determines the total amount of core storage available in the system and subtracts from this the amount of core used for non-overlayable MCP program segments and tables. This number is then multiplied by the Multiprocessing Factor (nominally set to 1) to determine how much core space should be considered to be available for scheduling purposes.

The Multiprocessing Factor can be changed by the operator so that the MCP thinks it has more, or less, core storage to use for running object programs. On a system containing eight memory modules, for example, the amount of core available for object programs is approximately 28,800 words. If the operator chooses to set the Factor to 0.8, the MCP considers that only 23,040 words are available to object

programs. Thus, the jobs which can run together have core requirements which total less than 23,040 words, and any other jobs are left in the schedule until one of the jobs in the mix comes to End-of-Job.

If, on the other hand, the Factor has been set to 1.2, the MCP considers that 34,560 words are available to user programs and continues to introduce jobs into the mix until the sum of their core requirements approaches 34,560 words.

In effect, the Multiprocessing Factor allows the operator to increase or decrease the tendency of the MCP to multiprocess. By increasing the Factor, the MCP can load more programs to be multiprocessed together; by decreasing the Factor, the MCP cannot load as many programs to be multiprocessed.

Experience has shown that a Factor of 1 causes the MCP to make quite reasonable decisions as to which jobs should be multiprocessed. It is conceivable that the Factor might be set as high as 1.5, or as low as 0.8, to effect better performance at some particular installation. Factors outside of this range are not suggested. In particular, a Factor greater than 2 almost certainly causes the MCP to load every job presented to it, causing the system to run out of memory if job requests are presented indiscriminately. A Factor of 0 (zero) causes only one job to be run at a time, with the feature that ZIPPed programs are not loaded until the calling program reaches End-of-Job.

Changing the Factor causes the MCP to "remember" the new value until it is changed. A Halt/Load does not reset the Factor.

SECTION 4
CONTROL INFORMATION

GENERAL.

Information about the various functions to be performed by the B 5500 is normally entered into the system via punched cards or typewriter messages. These cards, commonly called the schedule deck, are of two basic types: control cards and program-parameter cards. It is necessary to note that there are other cards that may appear in the schedule deck, notably source-deck cards and \$ sign cards.

This section describes in detail the control cards, the program-parameter cards, and the compiler-option cards. For typewriter messages, refer to Appendix C.

CONVENTIONS.

In describing the format of control information, the following conventions are used:

- a. If a word in a description is a word that would appear as part of the control information, this word is in UPPER CASE letters.
- b. If a word in a description is merely a word used in the expression of the description, the word is in lower case letters.
- c. If a word or phrase in a description has a particular definition, the word or phrase is contained within broken brackets; i.e., $\langle \ \rangle$.

DEFINITIONS.

A number of terms are used throughout the discussion of control information. Although many of these terms are well known, they are defined in order to avoid misinterpretation and to provide meanings for any persons unfamiliar with the terms.

<u>Term</u>	<u>Definition</u>
letter	Any letter of the alphabet of the English language.
digit	Any one of the characters listed below: 0 1 2 3 4 5 6 7 8 9
integer	A contiguous string of digits.
space	One or more blank card columns, or one or more horizontal space movements on a typewriter, whichever is relevant.
special character	Any one of the characters listed below: [≠ . \$ @] ≤ , ← - (≥ : + &) < " x % = > * # /
legitimate character	A letter, a digit, a special character, or a single space.
illegitimate character	Any card code that does not represent a legitimate character. When used in descriptions of control cards and

TermDefinition

	program-parameter cards, an illegitimate character is represented as a question mark (?).																																																
quote	The character ".																																																
string	Any contiguous string of legitimate characters, excluding quotes, that is preceded by and followed by a quote.																																																
identifier	Any contiguous string of letters and/or digits that begins with a letter and that is not greater than 63 characters in length.																																																
reserved character	Any one of the five characters listed below: - . = , ;																																																
reserved word*	Any one of the words listed below: <table> <tr> <td>ALGOL</td> <td>EU</td> <td>LOAD</td> <td>SERIAL</td> </tr> <tr> <td>BACK</td> <td>EXECUTE</td> <td>NO</td> <td>SLOW</td> </tr> <tr> <td>CC</td> <td>FAST</td> <td>PAPER</td> <td>SPECIAL</td> </tr> <tr> <td>CHANGE</td> <td>FILE</td> <td>PRINT</td> <td>SPO</td> </tr> <tr> <td>COBOL</td> <td>FORM</td> <td>PRIORITY</td> <td>STACK</td> </tr> <tr> <td>COMMON</td> <td>FORTRAN</td> <td>PROCESS</td> <td>SYNTAX</td> </tr> <tr> <td>COMPILE</td> <td>FREE</td> <td>PUBLIC</td> <td>TAPE</td> </tr> <tr> <td>CORE</td> <td>FROM</td> <td>PUNCH</td> <td>TO</td> </tr> <tr> <td>DATA</td> <td>INFO</td> <td>RELEASE</td> <td>UNIT</td> </tr> <tr> <td>DISK</td> <td>IO</td> <td>REMOVE</td> <td>UPDATE</td> </tr> <tr> <td>DUMP</td> <td>LABEL</td> <td>RUN</td> <td>USE</td> </tr> <tr> <td>END</td> <td>LIBRARY</td> <td>SAVE</td> <td>USER</td> </tr> </table>	ALGOL	EU	LOAD	SERIAL	BACK	EXECUTE	NO	SLOW	CC	FAST	PAPER	SPECIAL	CHANGE	FILE	PRINT	SPO	COBOL	FORM	PRIORITY	STACK	COMMON	FORTRAN	PROCESS	SYNTAX	COMPILE	FREE	PUBLIC	TAPE	CORE	FROM	PUNCH	TO	DATA	INFO	RELEASE	UNIT	DISK	IO	REMOVE	UPDATE	DUMP	LABEL	RUN	USE	END	LIBRARY	SAVE	USER
ALGOL	EU	LOAD	SERIAL																																														
BACK	EXECUTE	NO	SLOW																																														
CC	FAST	PAPER	SPECIAL																																														
CHANGE	FILE	PRINT	SPO																																														
COBOL	FORM	PRIORITY	STACK																																														
COMMON	FORTRAN	PROCESS	SYNTAX																																														
COMPILE	FREE	PUBLIC	TAPE																																														
CORE	FROM	PUNCH	TO																																														
DATA	INFO	RELEASE	UNIT																																														
DISK	IO	REMOVE	UPDATE																																														
DUMP	LABEL	RUN	USE																																														
END	LIBRARY	SAVE	USER																																														

* A reserved word is not recognized as such if it appears as a string. The same is true of a reserved character.

<u>Term</u>	<u>Definition</u>
comment	Any list of legitimate characters that does not contain either reserved words or reserved characters. A comment can consist entirely of spaces.
separator	An identifier, providing it is not a reserved character, a reserved word, a string, or a special character.
program identifier	An identifier of seven or less characters, excluding reserved words, or a string of seven or less characters.*
program suffix	An identifier of seven or less characters, excluding reserved words, or a string of seven or less characters.*
program name	Program identifier or program identifier suffix.
program specifier	Program identifier, separator, or program identifier suffix.
file identification	An identifier of seven or less characters excluding reserved words, or a string of seven or less characters.*
file identification prefix	An identifier of seven or less characters excluding reserved words, or a string of seven or less characters.*

* A program identifier, a program identifier suffix, a file identifier, or a file identifier prefix may appear in control information as more than seven characters; however, when this is the case, only the first seven characters are taken to be significant.

<u>Term</u>	<u>Definition</u>
multiple file identification	A file identification prefix.
data file name	A file identification prefix/file identification, or a file identification, in which case the file identification prefix is assumed to be zeros.
data file specifier	A file identification prefix/file identification.
data file designator	A data file specifier, or a file identification, in which the file prefix is assumed to be zeros.
file list	A file specifier, or a file list, file specifier.
change element	A <program specifier> <separator> <program specifier>, or a <data file specifier> <separator> <data file specifier>.
change list	A change element, or a change list, change element.
rdc	A term with one of the following three formats: ,rrr ,rrr, ddddd ,rrr, ddddd, cc where rrr is an integer of three or less characters and specifies the reel number, ddddd is a five character integer - its first two digits specify

Term

Definition

	the number of the day of the year; and cc is an integer of two or less characters and specifies the cycle number.
compiler name	COBOL, ALGOL, or FORTRAN, with <identifier>/DISK.
library tape name	multiple file identification.
unit mnemonic	Any one of the three-character codes listed below. The definition of each unit mnemonic, as recognized by the MCP, is listed to the right of the code.

Code

Definition

MTA	Magnetic tape unit A
MTB	Magnetic tape unit B
MTC	Magnetic tape unit C
MTD	Magnetic tape unit D
MTE	Magnetic tape unit E
MTF	Magnetic tape unit F
MTH	Magnetic tape unit H
MTJ	Magnetic tape unit J
MTK	Magnetic tape unit K
MTL	Magnetic tape unit L
MTM	Magnetic tape unit M
MTN	Magnetic tape unit N
MTP	Magnetic tape unit P
MTR	Magnetic tape unit R
MTS	Magnetic tape unit S

Term

Definition

Code

Definition

MTT	Magnetic tape unit T
MTX	All scratch tapes
CRA	Card reader A
CRB	Card reader B
LPA	Line printer A
LPB	Line printer B
CPA	Card punch A
PPA	Paper tape punch unit A
PPB	Paper tape punch unit B
PRA	Paper tape reader A
PRB	Paper tape reader B
SPO	Supervisory printer
CDA through CDZ, excluding CDI and CDO, and CD2 through CD9	Pseudo card readers A through Z, excluding I and O, and 2 through 9
DCA	Data communications control unit
DKA	Disk control A
DKB	Disk control B

system mnemonic

Any one of the 3-character codes listed below. The definition of each system mnemonic, as recognized by the MCP, is listed to the right of the code.

Code

Definition

SYA	System A (also referred to as O or blank*)
-----	--

*Pertains to those messages and entries concerning system identification when not under Sharedisk.

Term

Definition

Code Definition

SYB	System B (also referred to as 1)
SYC	System C (also referred to as 2)
SYD	System D (also referred to as 3)

aggregate file name = / = or = / file identification or = / program identification suffix, or file identification prefix / = or program identification / =.

aggregate file list File list, or aggregate file name, or aggregate file list, file list, or aggregate file list, aggregate file name.

MCP module list A list of the module options set TRUE when the current MCP version was compiled.

CONTROL INFORMATION VIA PUNCHED CARDS.

As stated previously, the punched cards used to supply control information to the MCP are classified as control cards and program parameter cards. These cards are distinguishable from other punched cards used in the system in that they are required to have an invalid character in column 1.

Aside from the invalid character in column 1, the information on control cards and program-parameter cards is in a free-field format, with the exception of label cards. Label cards have a fixed format. Although the information on control cards is in a free-field format, it must appear in the order denoted by the following descriptions.

Only the first 72 columns of a control card (with the exception of the label card) or label equation card may contain control information. The MCP ignores the information in columns 73-80. Also, if

the special character . (period) appears in a control card or label equation card, and is not a part of a <string>, all information following the period is ignored.

In the following descriptions (control cards, program-parameter cards, etc.), the information specified as being contained in a particular card may, in fact, be contained in more than one card, except in the case of a label card. If it is desired to continue information from one card to another, the character following the information which is continued must be a hyphen (i.e., the reserved character -). The hyphen cannot, however, divide an <identifier>. Also, the cards onto which the information is continued must not contain an <invalid character> in column 1.

Through the use of the semicolon (i.e., the reserved character ;), one punched card can contain information for more than one control card and/or program-parameter card. The appearance of a semicolon in a control card or program-parameter card denotes the logical end of that card and the beginning of another. When the semicolon convention is used, a question mark is neither required nor accepted in the control information following.

CONTROL CARDS.

In the following paragraphs, each control card used in the system is listed together with a detailed function of each card.

COMPILE CARD.

The COMPILE card is used to call out the Compiler for the purpose of compiling a source program. The COMPILE card designates the compiler to be used and the type of compile run to be made. A compile-and-go run, a compile-for-library run, or a compile-for-syntax-check run may be specified. Each of these three runs is described in the following paragraphs.

COMPILE-AND-GO RUN. This run causes the given program to be scheduled to run after an error-free compilation, but does not enter the program in the disk directory. The disk space used by the program is returned after the program is run. The COMPILE card for a compile-and-go run must contain the following information.

```
? COMPILE <program specifier> <comment> <compiler name>
      <comment>
```

COMPILE-FOR-LIBRARY RUN. This run causes the given program to be left on the disk and entered in the disk directory after an error-free compilation, but it does not cause the program to be scheduled to run. The COMPILE card for a compile-for-library run must contain the following information:

```
? COMPILE <program specifier> <comment> <compiler name>
      <comment> LIBRARY <comment>
```

Any label equation or program-parameter cards presented when a program is compiled to the library are retained and used for the execution of the program. The effect of these cards is overridden by other cards presented at execution time.

Suppose, for example, it is desired that a certain library program should have a priority of 4 whenever it is executed. This effect can be achieved by including a PRIORITY card when the program is compiled for the library.

The effect of this PRIORITY card is overridden by presenting another card at execution time. An EXECUTE callout, which includes a card which changes the priority to 2, will cause that execution to run with priority 2, but will have no effect on subsequent executions.

COMPILE-FOR-SYNTAX-CHECK RUN. This run causes no execution or directory action after the compilation. The COMPILE card for a compile-for-syntax-check run must contain the following information:

```
? COMPILE <program specifier> <comment> <compiler name>
           <comment> SYNTAX <comment>
```

Examples:

```
? COMPILE JOB BY CMB WITH ALGOL FOR LIBRARY
? COMPILE PATSER BY RNF ALGOL SYNTAX
? COMPILE "124-6A"/VEW USING ALGOL
? COMPILE PF SEPARATE "10" ALGOL LIBRARY
? COMPILE BRUTE BY BILL IN COBOL. GO GO GO
```

A compiler may have any [multi-file identification]. The [file identification] must be DISK.

For example, the control card

```
? COMPILE, ABC/DEF WITH GARBAGE; END
```

would cause program ABC/DEF to compile using a compiler named GARBAGE/DISK.

The word WITH must be present. The first identifier following the word WITH will be treated as the [multi-file identification] of the compiler. The word WITH is not necessary if the compiler's [multi-file identification] is ALGOL, COBOL, or FORTRAN.

EXECUTE CARD.

The EXECUTE card is used to call out a library program from the disk for execution. The EXECUTE card must contain the following information:

```
? EXECUTE <program specifier> <comment>
```

Examples:

```
? EXECUTE JOB/CMB
? EXECUTE PF NUMBER 10
? EXECUTE PROSORT BY IRP. THIS IS RUN 2.
```

REMOVE CARD.

The REMOVE card causes the specified file(s) to be removed from the disk directory and causes the disk space used by the file(s) to be made available for other use.

NOTE

The files SYSTEM/LOG and DIRCTRY/DISK cannot be removed from the disk.

To specify what files are to be removed, file lists and/or aggregate file names (i.e., an <aggregate file list>) are used. A file list, of course, provides a file specifier for each file to be removed. An aggregate file name specifies one or more files which are to be removed. Specifically, the various choices for REMOVE card information are explained as follows.

- a. If a REMOVE card contains information of the form:

? REMOVE <file list>

then all files that are both in the file list and the disk directory are removed.

- b. If a REMOVE card contains information of the form:

? REMOVE = / <file identification>

or

? REMOVE = / <program identification suffix>

then all files with the given file identification or program identification suffix, which are on the disk, are removed.

- c. If a REMOVE card contains information of the form:

? REMOVE <file identification prefix> / =

or

? REMOVE <program identification> / =

then all files with the given file identification prefix or program identification, which are on the disk, are removed.

d. If a REMOVE card contains information of the form:

```
? REMOVE = / =
```

then all files on the disk are removed.

To summarize, the information on a REMOVE card must contain the following information:

```
? REMOVE <aggregate file list>
```

Examples:

```
? REMOVE PROG/CAL, PROG/IRA, DATA/IRACAL
```

```
? REMOVE = / IRA
```

```
? REMOVE PROG / =
```

```
? REMOVE = / =
```

```
? REMOVE P/I, = / DISK, DATA / =, A/B
```

DUMP CARD - UNLOAD CARD.

The DUMP or UNLOAD card causes one or more library programs and/or data files to be copied on a scratch tape from disk. The file information written on the magnetic tape forms a multi-file reel, and is referred to as a library tape. DUMP and UNLOAD requests run with a MIX number as a program named LIBMAIN/DISK. LIBMAIN/DISK may be treated like any normal state program (i.e., DS-ed, ST-ed, etc.) with the exception that a UL message must be given in response to the # NO FILE message.

The DUMP card facility does not remove files from the disk directory. The UNLOAD card removes the specified file from the directory after the file has been dumped. The number of files to be dumped on a library tape may be specified. The maximum is 511 files. The MCP will start a sufficient number of LIBMAIN/DISK programs to dump all the files specified.

```
? DUMP 240 TO SYSTEM = / DISK; END
```

If 362 files with a <file-identification> of "DISK" were on the disk, two copies of LIBMAIN/DISK could be run at the same time if core requirements allow. The tape labeled SYSTEM would contain 240 files, and 122 files would be dumped to a tape labeled SYSTEM1.

NOTE

The files SYSTEM/LOG and DIRCTRY/
DISK cannot be dumped.

To specify what files are to be dumped, file lists and/or aggregate file names (i.e., an <aggregate file list>) are used. A file list, of course, provides a file specifier for each file dumped. An aggregate file name specifies that one or more files are dumped. Specifically, the various choices for DUMP card information are explained as follows:

- a. If a DUMP card contains information of the form:

? DUMP TO <library tape name> <file list>

then all files both in the file list and the disk directory are dumped.

- b. If a DUMP card contains information of the form:

? DUMP TO <library tape name> = / <file identification>

or

? DUMP TO <library tape name> = / <program
identification suffix>

then all files with the given file identification or program identification suffix, which are on the disk, are dumped.

- c. If a DUMP card contains information of the form:

? DUMP TO <library tape name> <file identification
prefix> / =

or

```
? DUMP TO <library tape name> <program
  identification> / =
```

then all files with the given file identification prefix or program identification, which are on the disk, are dumped.

- d. If a DUMP card contains information of the form:

```
? DUMP TO <library tape name> = / =
```

then all files on the disk are dumped.

- e. If a DUMP card contains information of one of the following forms:

```
? DUMP ACCESSD TO <library tape> <file list>
? DUMP <integer> ACCESSD TO <library tape> <file list>
? UNLOAD ACCESSD TO <library tape> <file list>
? UNLOAD <integer> ACCESSD TO <library tape> <file list>
```

then, only newly created disk files and disk files which have been accessed, except program files, since the last library LOAD, are dumped.

To summarize, the information on a DUMP or UNLOAD card must contain the following information:

```
? DUMP <separator> <library tape name> <aggregate file
  list>
```

Examples:

```
? DUMP TO IRASYST PROG/CAL, PROG/IRA, DATA/IRACAL
? DUMP TO IRASYST = / IRA
? DUMP TO CALSYST PROG / =
```

```

? DUMP TO LIBTAPE =/=
? DUMP TO LIBTAPE P/I, =/DISK, DATA/=, A/B
? UNLOAD TO SYSTEM =/=
? DUMP 240 TO SYSTEM =/DISK
? DUMP ACCESSD TO LIB =/USERTS
? UNLOAD ACCESSD TO LIB =/USERTS
? DUMP 173 ACCESSD TO LIB =/TSUSER
? UNLOAD 173 ACCESSD TO LIB =/TSUSER

```

LOAD CARD - ADD CARD.

The LOAD Card causes the files specified by <library tape name> and <file list> to be loaded on the disk and entered in the Disk Directory. The ADD Card loads the specified files only if the files are not already in the Directory. The LOAD or ADD requests run with a mix number as a program named LIBMAIN/DISK. LIBMAIN/DISK may be treated as any normal state program (i.e., DSed, STed, etc.) with the exception that a UL Message must be given in response to the # NO FILE Message.

NOTE

The files SYSTEM/LOG and
DIRCTRY/DISK cannot be loaded.

A variation of the LOAD Construct also makes it possible for files to be loaded to a specified disk file electronics unit (EU) or a specified "speed of disk." Examples are as follows:

```

CC LOAD TO EU<integer> FROM <label> =/=; END
CC LOAD TO FAST FROM <label> =/=; END
CC LOAD TO SLOW FROM <label> =/=; END

```

If the specified EU or SPEED is not available, standard-library-maintenance, no-user-disk action is taken.

To specify what files are to be loaded, file lists and/or aggregate file names (i.e., an <aggregate file list>) are used. A file list provides a file specifier for each file to be loaded. An aggregate

file name specifies one or more files are to be loaded. Specifically, the various choices for LOAD Card information are explained as follows:

- a. If a LOAD Card contains information of the form:

? LOAD FROM <library tape name> <file list>

all files that are both in the file list and the library tape are loaded.

- b. If a LOAD Card contains information of the form:

? LOAD FROM <library tape name> =/<file identification>

or

? LOAD FROM <library tape name> =/<program identification suffix>

all files with the given file identification or program identification suffix, which are on the library tape, are loaded.

- c. If a LOAD Card contains information of the form:

? LOAD FROM <library tape name> <file identification prefix>/=

or

? LOAD FROM <library tape name> <program identification>/=

all files with the given file identification prefix or program identification, which are on the library tape, are loaded.

- d. If a LOAD Card contains information of the form:

? LOAD FROM <library tape name> =/=

all files on the library tape are loaded.

To summarize, the information on a LOAD or ADD Card must contain the following information:

```
? LOAD <separator> <library tape name>
      <aggregate file list>
```

Examples

```
? LOAD FROM IRASYST PROG/CAL, PROG/IRA, DATA/IRACAL
? LOAD FROM IRASYST =/IRA
? LOAD FROM CALSYST PROG/=
? LOAD FROM LIBTAPE =/=
? LOAD FROM LIBTAPE P/I, =/DISK, DATA/=, A/B
? ADD FROM SYSTEM =/A, B/=
```

CHANGE CARD.

The CHANGE Card is used to change the names of program files and/or data files which are on disk. The first <file specifier> in a <change element> signifies the name of the file whose name is to be changed. The second <file specifier> in a <change element> signifies the new name for the file. The CHANGE Card must contain the following information:

```
? CHANGE <change list>
```

Examples

```
? CHANGED DATA1/"001" TO DATA1/"0001"
? CHANGE ALGOL/DISK TO OLDALGOL/A, NEW ALGOL/A TO ALGOL/DISK
? CHANGED A BY ME TO B BY ME, AB/C TO AC/B, BAD/N TO GOOD/N
```

LABEL CARD.

The LABEL Card may be used to relate a card file with a <data file name> and other label information. With the exception of the <invalid character> required in column 1, the information in a LABEL Card is defined as it is for a standard system label, as would be used on a magnetic tape file.

A LABEL Card has a fixed format as described below. In the description below, the character b signifies a single <space>.

<u>Characters</u>	<u>Field Description</u>
1-8	Must contain ?LABELbb
9	Must be 0 (zero)
10-16	Multiple file identification
17	Must be 0 (zero)
18-24	File identification
25-27	Reel number
28-32	Creation date
33-34	Cycle number
35-64	(Irrelevant for card files)
65-72	User's portion (for COBOL)

DATA CARD.

The DATA Card can be used in lieu of a LABEL Card if the <multiple file identification> for a card file consists of zeros and if there is no desire to provide label information other than a <file identification>. The DATA Card must contain the following information:

? DATA <file identification>

Example

? DATA CARDS

SET ACCESSD OR RESET ACCESSD CARD.

The SET ACCESSD or RESET ACCESSD Card is used to set or reset the flag which is set when a disk file has been accessed.

Whenever a program write-accesses a permanent file, the MCP sets the file access bit of the disk file header. This bit is reset only when the file is loaded by LIBRARY LOAD or by the RESET Control Card. By this implementation, only accessed files are dumped by the DUMP ACCESSD Control Card.

The SET ACCESSD or RESET ACCESSD Card must contain the following information:

? SET ACCESSD <file list>
or
? RESET ACCESSD <file list>

Examples

? SET ACCESSD A/B
? RESET ACCESSD A/B,C/D,Y/Z

END CARD.

The primary function of the END Card is to denote the end-of-card information for a particular program. This designation of the end of information is required by the MCP whenever a program is terminated for any reason while it has card information yet to be read. Consequently, the END Card is required to be the last card in a deck pertaining to a program. Thus, the END Card relevant to a particular program should normally not be followed by any other control card, program-parameter cards, or any data cards for that same program.

If a program attempts to read an END Card as data, an end-of-file notification occurs. However, the END Card is not required to denote the end of a data file. An attempt to read any control card as data causes an end-of-file notification. Consequently, if a program requires more than one card file, the end of one file is denoted by the LABEL Card or DATA Card for the next.

The END Card must contain the following information:

? END <comment>

Example

? END

END CONTROL CARD.

The END CONTROL Card is used to denote the end of a CONTROL DECK File used by the system program LDCNTRL/DISK. This card serves to

discontinue card reader input to a disk, pseudo-reader file and causes LDCNTRL/DISK to go to End-of-Job.

NOTE

This does not affect those card decks previously read into the pseudo reader; they continue to execute until the pseudo reader is turned off (set to zero).

The END CONTROL Card must contain the following information:

? ~~END CONTROL~~ ? END PACKETS
see separate #10, APPX B, p 17

Example

? END CONTROL

PROGRAM-PARAMETER CARDS.

Program-parameter cards are classified as program-parameter cards for compilers and program-parameter cards for object programs. The only difference between the two classifications is that the cards for compilers have a <compiler name> following the <invalid character> and refer to the specified compiler; the cards for object programs do not contain a <compiler name> and refer to the object program.

Program-parameter cards for a compiler are included in the program-parameter cards that immediately follow a COMPILE Card. Also, for a compile-and-go or a compile-for-library run, the program-parameter cards for the object program are included in the program-parameter cards immediately following the COMPILE Card. For compile-for-syntax check, program-parameter cards may be included as for a compile-and-go run, but they are ignored.

When an object program is called for execution through use of an EXECUTE Card, the program-parameter cards for the object program

must immediately follow the EXECUTE card. Within a group of program-parameter cards, order is irrelevant.

In the paragraphs that follow, each program-parameter card of the B 5500 System is listed, together with the function of each card.

PROCESS CARD.

The PROCESS card specifies the maximum amount of processor time an object program or compiler is allowed. If the processor time for the program concerned exceeds the amount specified, the MCP will terminate the program. The \langle integer \rangle on the PROCESS card specifies the maximum processor time in minutes. The \langle integer \rangle on the process card must not contain more than eight digits.

The PROCESS card for a compiler must contain the following information:

```
?  $\langle$ compiler name $\rangle$  PROCESS  $\langle$ comment $\rangle$  =  $\langle$ integer $\rangle$ 
```

Example:

```
? ALGOL PROCESS TIME = 1  
? COBOL PROCESS = 3
```

The PROCESS card for an object program must contain the following information:

```
? PROCESS  $\langle$ comment $\rangle$  =  $\langle$ integer $\rangle$ 
```

Example:

```
? PROCESS = 4  
? PROCESS MAXIMUM = 5
```

IO CARD.

The IO card specifies the maximum amount of I/O channel (i.e., I/O control unit) time an object program or compiler is allowed. If the I/O channel time for the program exceeds the amount specified, the

MCP will terminate the program. The $\langle \text{integer} \rangle$ on the IO card specifies the maximum channel time in minutes. The $\langle \text{integer} \rangle$ contained on an IO card must not contain more than eight digits.

The IO card for a compiler must contain the following information:

? $\langle \text{compiler name} \rangle$ IO $\langle \text{comment} \rangle$ = $\langle \text{integer} \rangle$

Examples:

? ALGOL IO = 4

? COBOL IO TIME MAX = 5

The IO card for an object program must contain the following information:

? IO $\langle \text{comment} \rangle$ = $\langle \text{integer} \rangle$

Examples:

? IO = 4

? IO MAXIMUM ALLOWED = 3. COMMENT

STACK CARD.

The STACK card specifies the number of words assigned in core memory for the stack of the compiler or the object program. The MCP assigns 512 words unless a STACK card specifies otherwise. The $\langle \text{integer} \rangle$ in the STACK card specifies the stack size in B 5500 words.

The STACK card for a compiler must contain the following information:

? $\langle \text{compiler name} \rangle$ STACK $\langle \text{comment} \rangle$ = $\langle \text{integer} \rangle$

Examples:

? ALGOL STACK = 320

? COBOL STACK SIZE = 640

The STACK card for an object program must contain the following information:

? STACK <comment> = <integer>

Examples:

? STACK SHALL BE = 256

? STACK = 560

PRIORITY CARD.

The PRIORITY card specifies the priority assigned a compiler or object program. Priorities may range from 0 to 32767, where 0 is the highest priority and 32767 is the lowest. For scheduling purposes, priorities greater than MIXMAX; i.e., priorities greater than the maximum number of programs allowed in the MIX, are taken to be equivalent to MIXMAX. However, during processing, the specified priorities are used, regardless of value. The MCP assigns a priority of (MIXMAX+1)DIV2 unless a priority card specifies otherwise. The <integer> on the PRIORITY card specifies the priority assigned.

The PRIORITY card for a compiler must contain the following information:

? <compiler name> PRIORITY <comment> = <integer>

Examples:

? ALGOL PRIORITY FOR THIS RUN = 1

? COBOL PRIORITY = 3

The PRIORITY card for an object program must contain the following information:

? PRIORITY <comment> = <integer>

Examples:

? PRIORITY = 0

? PRIORITY LOWEST I AM ALLOWED = 4

FILE CARD (LABEL EQUATION).

The FILE card, often referred to as the label equation card, may specify the <data file name> to be associated with a file identifier used in a source program to refer to a particular file. Also, the FILE card can specify various options for output files.

In the following description, the term <file identifier> refers to the file identifier used in I/O statements in the source program. The terms described in the following paragraphs are optional; i.e., they may or may not be used. However, if they are included, they must occur in the order specified.

- a. The term <rdc> may be included.
- b. The term forms option may be used to cause the MCP to notify the system operator before a file requiring special forms is opened. The following word specifies <forms option> : FORM.
- c. The term no-label option may be used to notify the MCP not to place a label on the specified file. The following words are used to specify the <no-label option> : NO LABEL.
- d. The term <output medium> may be used to specify the type of output to be used for the file.
 - 1) The <output medium> BACKbUP can specify that a line-printer file is to be placed on a printer backup file.
 - 2) The <output medium> PRINT OR BACKbUP can specify that a line-printer file can be placed on a printer backup file if a line printer is not available.
 - 3) The <output medium> SPECIAL denotes that the first three characters of the <file identification> specify the <unit mnemonic> for the output unit to be used for the file. *Will override PBDONLY*

- 4) If <output medium> DISK is specified, a random accessing technique is assumed. SERIAL and UPDATE refer to disk files with serial and update accessing techniques.
- 5) The <output medium> terms, DISK, SERIAL, UPDATE, or TAPE, can also be used to equate input files to disk or tape.
- 6) The remaining choices for <output medium> are self-explanatory. If no <output medium> is specified, magnetic tape (i.e., TAPE) is assumed.
- 7) Any of the following reserved words are used for <output medium>:

DISK	BACKbUP
TAPE	PRINT OR BACKbUP
PUNCH	SERIAL
PAPER TAPE	UPDATE
PRINT	SPO
	SPECIAL

The information on a FILE card for a compiler must appear as follows:

? <compiler name> FILE <file identifier> = <data file designator> <rdc> <forms option> <no-label option> <output medium>

Examples:

? ALGOL FILE CARD = MFID/FID SERIAL
 ? ALGOL FILE TAPE = ZILCH
 ? ALGOL FILE LINE = LINE BACKbUP

The information on a FILE card for an object program must appear as follows:

? FILE <file identifier> = <data file designator> <rdc> <form option> <no-label option> <output medium>

Examples:

? FILE REED = MULFILE/FILEID
 ? FILE RITE = MUL/FID NO LABEL TAPE


```
? FILE RIGHT = L/PRNT FORM PRINT
? FILE ABC = "001"/KZ00,01,64350, 02
? FILE XYZ = M/N, 3, 64351
? FILE F1 = MTEFX01 SPECIAL
```

COMMON CARD.

The COMMON card may specify an \langle integer \rangle which is converted to binary and stored as an integer in the PRT (Program Reference Table) of an object program or compiler, just before the object program or compiler is initiated. The PRT cell in which the integer is stored is the first cell beyond cell 20, which contains a zero at load time. PRT cells reserved for simple variables, arrays, and various other program entities, contain zeros at load time. Consequently, when the COMMON card is used, the variable which is to receive the value of the \langle integer \rangle should occur as the first identifier declared in the program.

The \langle integer \rangle contained on a COMMON card must not contain more than eight digits. There is at most one COMMON card per object program or compiler.

The COMMON card for a compiler must contain the following information:

```
?  $\langle$ compiler name $\rangle$  COMMON  $\langle$ comment $\rangle$  =  $\langle$ integer $\rangle$ 
```

Examples:

```
? ALGOL COMMON = 127
```

The COMMON card for an object program must contain the following information:

```
COMMON  $\langle$ comment $\rangle$  =  $\langle$ integer $\rangle$ 
```

Example:

```
? COMMON = 12345678
```

THE UNIT CARD.

The UNIT card may be used to relate a ^a <data file name> to a particular I/O unit. It is required when an input file does not have a label and the IL <unit mnemonic> is not to be used.

The UNIT card must contain the following information:

? UNIT <unit mnemonic> = <data file designator>

or

? UNIT <unit mnemonic> = <data file specifier> <rdc>

Examples:

? UNIT MTE = UNLAB/FIEL

? UNIT MTA = FILEID, 2, 66262

THE CORE CARD.

The CORE card is a program-parameter card which allows the operator or programmer to override the compiler's estimate of the amount of core storage required for efficient execution of a program. The core requirement referred to here is effective only in deciding when the program should be multiprocessed with other programs.

This card need be used only when it is noticed that the compiler's estimate is unusually high or low. If it is noticed that the MCP has a general tendency to load a particular program when it probably should not have been loaded (i.e., the system bogs down unacceptably because of overlay after starting the job), then a CORE card should be used to increase the compiler-provided core requirement for that job.

The <integer> contained on the CORE card is the core requirement in B 5500 words, and is not greater than 32768. If the <integer> on the CORE card is greater than 32768, a value of 32768 is used instead.

The CORE card for the compiler must contain the following information:

? <compiler name> CORE <comment> = <integer>

Example:

? ALGOL CORE = 10000

The CORE card for an object program must contain the following information:

? CORE <comment> = <integer>

Example:

? CORE = 6000

THE SAVE CARD.

The SAVE card may be used when a program is being compiled to the library, and to specify how long the compiled program will be saved on disk without becoming expired. The names of all expired files may be obtained by using the EX message.

The <integer> on the SAVE card must not contain more than eight digits; it specifies the number of days the program will be retained.

A SAVE card for a compiler will have no effect and should not be used. The SAVE card for an object program must contain the following information:

? SAVE <comment> = <integer>

Example:

? SAVE PERIOD = 15 DAYS

COMPILER OPTION CARDS.

The Compiler Option cards are discussed in the following paragraphs.

\$ CARD.

The absence of this card specifies that the source program is entered as input via the card reader and listed. A further discussion of this card is covered separately below for ALGOL, COBOL, and ESPOL.

ALGOL SOURCE PROGRAMS.

The format of this card for use with ALGOL is free form except for the following conditions:

- a. The \$ is placed in column 1.
- b. The input medium option (i.e., CARD or TAPE) must precede all other options used.
- c. The options are delimited by at least one blank.

The syntax of the \$ Card for ALGOL is as follows:

$\langle \$ \text{ card} \rangle ::= \$ \langle \text{control options} \rangle$

$\langle \text{control options} \rangle ::= \langle \text{option} \rangle \mid \langle \text{option} \rangle \langle \text{control option} \rangle$

$\langle \text{option} \rangle ::= \text{CARD} \mid \text{TAPE} \mid \text{LIST} \mid \text{NEWbTAPE} \mid \text{NEW} \mid \text{PRT} \mid \text{DEBUGN} \mid$
 $\text{SGL} \mid \text{SINGLE} \mid \text{PUNCH} \mid \text{CHECK} \mid \langle \text{sequence option} \rangle \mid \text{VOID} \mid$
 $\langle \text{empty} \rangle \mid \text{VOIDT} \mid \text{SEQXEQ}$

$\langle \text{sequence option} \rangle ::= \text{SEQ} \langle \text{start} \rangle \langle \text{inc} \rangle \mid \text{SEQ} \langle \text{inc} \rangle$

$\langle \text{start} \rangle ::= \langle \text{integer} \rangle \mid \langle \text{empty} \rangle$

$\langle \text{inc} \rangle ::= + \langle \text{integer} \rangle \mid \langle \text{empty} \rangle$

CARD ::= The source program is entered as input via the card reader.

CHECK ::= The format of the ALGOL Control Card (\$ Card) provides a sequence-checking facility. The CHECK appearing anywhere on the \$ Card causes the ALGOL Compiler to check all input card images (from card reader, OCRDIMG Tapes, or CAST Tapes) for proper sequence order. If the next card to be compiled

has a lower sequence number than the previously compiled card, a warning message, SEQ, is printed to the right of the card image on the line printer. If the line printer is not open, it is opened to print the card image and the warning message. The message SEQ is only a warning and is not considered an error by the Compiler.

TAPE ::= The source program is entered as input via magnetic tape. Updating source program cards is accepted from the card reader.

LIST ::= The source program is listed. In addition, the segment number and the relative address of the last word generated for each source program statement appear to the right of the sequence number of the statement.

NEWbTAPE or NEW ::= Generates OCRDIMG Tape.

PRT ::= The relative PRT locations of the program assigned by the Compiler are printed immediately following, and to the left of, the associated source program statement.

DEBUGN ::= The object code produced by the Compiler is listed following each ALGOL Statement.

PUNCH ::= If a source program card or card image is encountered which contains a syntactical error, the card is punched in its original form.

SINGLE or SGL ::= The source program listing produced by the LIST Option is single-spaced. This option has no effect if LIST is not also specified.

VOIDT ::= This option acts similarly to VOID with the exception that VOIDT specifically voids card images on the TAPE Input File and has no effect on other card images being read (compiled).

SEQXEQ ::= This is the time sharing option which is automatically set when running under the time sharing system from a remote

terminal. The option gives line numbers (card numbers) of syntax errors to the associated remote terminal.

SEQ ::=

- a. This option resequences the listing on the LINE File as well as any NEWTAPE File that is being made.
- b. The specifications following SEQ have the following interpretations:

⟨start⟩ - the sequence number to be assigned to the first card of the source program.

⟨inc⟩ - increment.

VOID ::= All records on the source file whose sequence numbers are equal to or greater than the VOID card sequence number and less than the restart sequence number will be skipped. The restart sequence number is from one to eight contiguous characters, excluding an invalid character, following \$ VOID.

COBOL PROGRAM.

The format for the \$ card for COBOL is now free form, except for the following conditions:

- a. The \$ must be placed in Columns 1 or 7. If it is in Column 7, then Columns 1 through 6 are assumed to be a Sequence number.
- b. The input media option (i.e., card or tape) once set need not be set every time and must occur, if present, immediately after the \$.
- c. The options are delimited by at least one blank.

The absence of any \$ card specifies that the source program is input via the card reader and that it is listed.

The syntax for the \$ card for COBOL is as follows:

```
<$ CARD> ::= <control options> | $ <options>
<control options> ::= CARD | TAPE | VOID | <options>
<options> ::= LIST | NEW TAPE | PRT | PUNCH | SPEC | DEBUGN |
            <sequence option> | <inc> | SINGLE | SGL
<sequence option> ::= SEQ <start> <inc> | SEQ <inc> | SEQ |
                  TSSREDIT | FREE
<start> ::= <integer> | <empty>
<inc> ::= + <integer> | <empty>
```

CARD ::= The source program is input via the card reader.

TAPE ::= The source program is input from another source; i.e., magnetic tape or disk.

Updating source program cards are accepted from the card reader.

VOID ::= The compiler will remove the source program card whose sequence number is identical to the one in columns 1-6 of this card. If there is a number after the VOID in the \$ card the voiding option will continue until that number is exceeded. If the number is less than six digits then it is left justified and zero filled on the right.

LIST ::= The source program is listed.

NEW TAPE ::= The source, independent of input, is placed on a magnetic tape.

The label of this tape is SOLT.

PRT ::= The program's relative PRT locations assigned by the compiler are listed. The paragraph number is also printed.

PUNCH ::= The source program, independent of input, is reproduced on the card punch.

SPEC ::= This will suppress MOVE TRUNCATION, SEQUENCE ERROR and corresponding messages.

DEBUGN ::= The object code produced by the compiler is listed.

SEQ ::= This option will resequence the source deck using the existing sequence number of this card (columns 1-5) and incrementing by 100. If an integer occurs after SEQ and before a + then this integer is assumed to be the new sequence number. The integer after a + is assumed to be the increment for resequencing. Zero suppression is assumed if an integer is less than six digits.

SINGLE or SGL ::= This option will cause the compiler to produce a single spaced listing.

A \$ TAPE card is used at any point in the card deck to initiate reading from the SOLT tape. A subsequent \$ CARD will suppress reading from the SOLT tape. Reading from the tape will begin again when another \$ TAPE card is encountered, starting with the record

following the last one processed before the \$ CARD was encountered. The card deck following the last \$ TAPE card must contain a sequence number higher than any on the SOLT tape (such as 999999), unless the last card contains END-OF-JOB in columns 8-17.

\$ CARDS which are not first in the deck must allow for sequence numbers. To do this, the \$ sign is put in column 7 instead of column 1 and the other option-fields. For example, CARD, LIST, and PRT are placed six columns to the right of the above \$ CARD layout.

\$ CARD FOR ESPOL.

ESPOL has two types of \$ cards: a \$ card which specifies I/O options, such as is used with the Extended ALGOL Compiler and a \$ VOID card.

The format of I/O option \$ cards for ESPOL is similar to that for \$ cards in Extended ALGOL. A \$ must appear in column 1 followed either by the word CARD or the word TAPE. The remainder of the card may contain any of the following option words in a free field format:

LIST
PRT
NEW TAPE
DECK [digit] [digit]
DEBUGN
STUFF
INTRINSIC
SGL
SINGLE

The words CARD and TAPE are used with the ESPOL Compiler for the same purpose as with the Extended ALGOL Compiler. CARD is used when the source program is on cards alone, and TAPE is used when compiling from tape using the card file as a patch deck. Also, the words LIST, PRT, NEW TAPE, SGL, SINGLE, and DEBUGN provide the same actions for ESPOL as they do for the Extended ALGOL Compiler. The word DECK, which is not an ALGOL option, is used to specify that the program generated by the compilation is punched on cards rather than written on disk. Once the DECK option is set, it is not revoked due to subsequent \$ cards which do not specify the option.

The use of the STUFF option causes a card to be punched with the following format for each procedure, array, or variable assigned to the PRT by the Compiler.

<u>Columns</u>	<u>Field Content</u>
1-4	Decimal class, where:
	10 = Procedure
	12 = Stream Procedure
	13 = Boolean Stream Procedure
	14 = Real Stream Procedure
	15 = Integer Stream Procedure
	17 = Boolean Procedure
	18 = Real Procedure
	19 = Integer Procedure
	21 = Boolean
	22 = Real
	23 = Integer
	25 = Boolean Array
	26 = Real Array
	27 = Integer Array
	30 = Name

<u>Columns</u>	<u>Field Description</u>
5-8	Decimal PRT address
9-80	Identifier, left justified, with blank fill

The production of these output cards is initiated by the appearance of the word STUFF on a \$ card. The introduction of another \$ card not containing the word STUFF will inhibit the punching of these cards.

The INTRINSIC option is turned on by the appearance of the word INTRINSIC on a \$ control card signifying to ESPOL that it is compiling an intrinsic file and enabling it to produce a disk file in the format expected by the MCP.

An intrinsic program (i.e., the symbolic input to ESPOL when using the \$INTRINSIC option) is subject to the following restrictions:

- a. The outer block of the program may contain only declarations; no statements are allowed.
- b. Only Non-Save Procedures may be declared. SAVE Procedures and Stream Procedures are prohibited.

NOTE

ESPOL uses a modified construct of the Stream Procedure which is referred to as the In-Line STREAM Statement.

- c. No SAVE array may be declared.
- d. Any variable declared in the outer block must be equated to an R or F relative address through use of a [relative address expression] in its declaration.

It is noted that the \$INTRINSIC option is required when compiling the symbolic file for the intrinsic of the programming system. When compiling these intrinsics, the ESPOL file DISK is equated to INT/DISK.

The \$ VOID card has the format \$ VOID [sequence number]; the \$ is in column one, and the [sequence number] is recognized as the field of eight consecutive characters starting with the first non-blank

character following the word VOID. A VOID card in an ESPOL patch deck causes cards on the symbolic input tape to be deleted from the compilation, and, if a NEW TAPE is being created during the compilation, the voided cards are not written on the new tape. Card images on a symbolic input tape which are affected by a VOID card are those with sequence numbers equal to or greater than the sequence number of the VOID card and less than the value in the [sequence number] field.

MCP MODULARITY.

The current MCP is modular in design, and it incorporates all functions available in previous standard MCP's.

Modularity is achieved by the inclusion of \$INCLUDE and \$OMIT cards. The parameters defined on these cards must be given values of TRUE or FALSE by using \$SET cards in the change deck at compile time. The compiled MCP is a function of the values given the parameters. It requires approximately 10 minutes to compile any MCP, regardless of the modules chosen.

Parameters for which \$SET cards must be included when compiling DCEPSY are:

<u>PARAMETER</u>	<u>DESCRIPTION</u>
BREAKOUT	If BREAKOUT is set TRUE, then code for the breakout-restart facility is included in the MCP. If set FALSE, the breakout-restart facility is not available.
DEBUGGING	If DEBUGGING is set TRUE, then all debugging facilities of the MCP are available including memory dump, console access to core memory, access to disk via the console, and trace. If DEBUGGING is set FALSE, then debugging facilities are unavailable with the exception noted below by the DUMP parameter.
DUMP	If DUMP is set TRUE, then the memory dump facility is provided via the console. If DUMP is set FALSE, this facility is not provided.

PARAMETER

DESCRIPTION

DUMP (cont)

NOTE

DEBUGGING set TRUE overrides the setting of the DUMP option since a system with full debugging facilities has memory dump capabilities via the console.

DFX

DFX set TRUE indicates that the user has a B 451 Expanded Disk File Control and two disk file control units. Code is compiled into the MCP to optimize disk I/O for this system configuration.

INQUIRY

If INQUIRY is set TRUE, code is included in the MCP for the data communications INQUIRY System. If INQUIRY is set FALSE, this code is omitted.

NOTE

An INQUIRY System precludes a DATACOM System and vice versa. The parameters INQUIRY and DATACOM should not both be set TRUE.

DATACOM

If DATACOM is set TRUE, code is included in the MCP for data-communications DATACOM II facilities. If DATACOM is set FALSE, this code is omitted.

NOTE

An INQUIRY System precludes a DATACOM System and vice versa. The parameters INQUIRY and DATACOM should not both be set TRUE.

DCSPO

If DCSPO is set TRUE, SPO facilities are available to the user at remote stations. If DCSPO is set FALSE, remote stations have no SPO capabilities.

<u>PARAMETER</u>	<u>DESCRIPTION</u>
DCSPO (cont)	<p style="text-align: center;">NOTE</p> <p>The DCSPO and DCLOG facilities are functions of a DATACOM System. DCSPO and DCLOG should be set FALSE if DATACOM is set FALSE.</p>
DCLOG	<p>If DCLOG is set TRUE, logging facilities for remote stations are compiled into the MCP. If DCLOG is set FALSE, this facility is not provided.</p> <p style="text-align: center;">NOTE</p> <p>The DCSPO and DCLOG facilities are functions of a DATACOM System. DCSPO and DCLOG should be set FALSE if DATACOM is set FALSE.</p>
DISKLOG	<p>If DISKLOG is set TRUE, logging facilities for disk files are compiled into the MCP. If DISKLOG is set FALSE, this facility is omitted.</p>
SAVERESULTS	<p>If SAVERESULTS is set TRUE, code is included in the MCP for the possible debugging of data communications I/O descriptors and result descriptors. If SAVERESULTS is set FALSE, the SAVERESULTS code is omitted.</p>
SHAREDISK	<p>If SHAREDISK is set TRUE, code is included in the MCP for the SHAREDISK System. If SHAREDISK is set FALSE, the SHAREDISK code is omitted. The SHAREDISK and BREAKOUT Options are mutually exclusive; hence, break-out-restart facilities are not available under SHAREDISK. The shared disk features enabled by this option are dependent on SHAREDISK hardware. This option cannot be used if the File Protect Memory Unit is not present.</p>
STATISTICS	<p>If STATISTICS is set TRUE, code is included in the MCP</p>

PARAMETER

DESCRIPTION

STATISTICS (cont) for maintenance of three additional pseudo logs.
If STATISTICS is set FALSE, the STATISTICS code
is omitted.

FORTRAN COMPILER.

The format of the Dollar Sign Card for FORTRAN is:

<u>Card Columns</u>	<u>Contents</u>
1	\$
2-72	Options in free field format
73-80	Card number or blank

The Dollar Sign Card is placed:

- a. Immediately after the MCP control cards used for compilation and immediately before the first FORTRAN FILE Card or FORTRAN Source or Patch Card if no FILE Cards are used.
- b. Anywhere else in the source or patch deck with a proper sequence number in order to change options at some point in compilation; e.g., to list only a part of the compiled source program.

Example:

	<u>Sequence Number</u>
\$CARD	00000100
...	...
...	...
...	...
A=B+C	00009000
\$CARD LIST	00009100
X=SQRT	00009200
PAR=TAN(X/A)	00009300
...	...
V=SIN(X+Y-Z)	00012200
\$CARD	00012300
...	...

Only cards 00009200 through 00012200 are listed on the file LINE.

When dollar sign cards are grouped together, all of them are ignored except for the last one. If no dollar sign card is included with the source deck, then the CARD and LIST options are assumed.

The various options available are as follows:

TAPE or CARD

- a. One of these, but not both, is the first option on the dollar sign card immediately following the dollar sign.
- b. CARD indicates to the compiler that the source program input is entirely from the file labeled CARD.
- c. TAPE indicates to the compiler that the source program input is from the file labeled TAPE and that change or patch cards may be input from the file labeled CARD. If a change or patch card file is used, then it is merged into the source program from the file labeled TAPE as a function of the sequence number in columns 73-80. If a

listing is obtained, then the source statements from the TAPE file will have a T following the sequence number, and the source statements being merged from the CARD file will have an R following the sequence number on the compiled source listing. The merging process uses the B 5500 alphanumeric collating sequence.

LIST

- a. If present, then a compiled source listing of the source program is made on the file LINE, including any change or patch cards.
- b. Segment and address information is also listed with the source program.

NEW or NEW TAPE

- a. If present, a new source tape file labeled FORSYM is created which includes all change or patch cards and FILE cards, but does not include dollar sign cards.

PRT

- a. If present, then a listing of the source program is made on the file LINE, including any change or patch cards. At the end of each program unit listing, a listing of PRT and stack assignments for each local identifier within that program unit is made.
- b. At the end of the entire program, PRT assignments for all global names are listed.
- c. If PRT is specified, then LIST is assumed.

DEBUGN

- a. If present, then the actual machine code emitted by the compiler is also listed on the file LINE together with octal values of constants and format of PRT entries.
- b. If DEBUGN is specified, then PRT and LIST are assumed.

TRACE

- a. If present, then information is listed on the file LINE which indicates how the FORTRAN compiler is analyzing the syntax of the source program.
- b. TRACE is used only in extreme cases because of the great volume of output produced.
- c. If TRACE is specified, the LIST, PRT, and DEBUGN options are assumed.

SEQ f s i

- a. If present, the listing on file LINE and the new source program on the file NEWTAPE, labeled FORSYM (if NEW or NEW TAPE is specified), are resequenced.
- b. The specifications following SEQ have the following interpretations:
 - f - the sequence number of the first card of the source program.
 - s - any special character, usually plus (+) or comma (,).
 - i - increment. If $i=0$, or i is not a number, then an increment of 1000 is used.
- c. The SEQ option, if used, is the last option on the dollar sign card.

HOL

- a. If the source cards are punched in IBM card code, then the HOL option need not be used. If this is the case, then the listing of the source program produced by the compiler is in IBM card codes; e.g., it is printed as %, = is printed as #, etc. However, the compiler will properly interpret the source program and compile it.
- b. If the source cards are punched in IBM card code and the HOL option is used, then all characters are converted to BCL before printing on the file LINE.
- c. If the source cards are punched in IBM/360 card code, then the HOL option must be used to convert the source program to BCL.
- d. The HOL option will translate all IBM or IBM/360 cards to BCL, including strings and Hollerith constants. This option also causes the object program produced by the compiler to automatically convert into BCL data read with an A format specification and data read into Hollerith strings.
- e. The use of the HOL option will slow compilation speed. For repeated compilations from large source programs, it is advantageous to use the NEW TAPE option with HOL on the first compilation. Thereafter, compilations are made without the HOL option from the generated source tape.

TIME

- a. If present and if the LIST option is not present, then the source program is not listed, but, at the end of the compilation, compilation information is listed on the file LINE.

CHECK

- a. If present and if TAPE and CARD files are being merged at compilation, then the sequence numbers in columns 73-80 of the two files are checked. If a record(s) from from the CARD file is not in sequence, then a warning message is printed on the file LINE:

SEQUENCE ERROR "n" < "p",

where n is the new sequence number and p is the old sequence number. The B 5500 alphanumeric collating sequence is used.

VOID n

- a. If present, VOID is the only option on the dollar sign card. This option is used only when merging a CARD and TAPE file.
- b. If the NEW option is previously specified, VOID is present, and n is blank, then the record on the TAPE file with the same sequence number (in columns 73-80) as the \$VOID card is ignored by the compiler. Also, it is not listed on the file LINE, and it is not inserted in the file NEWTAPE.

- c. If present, and if n is not blank, n must be the sequence number of a record existing in the TAPE file and, in addition, the \$VOID card must have a sequence number in columns 73-80. The records in the TAPE file, starting with the record which has the same sequence number as the \$VOID card (columns 73-80), are ignored up to but not including the record in the TAPE file with the sequence number n. These records are ignored by the compiler, not listed in the file LINE, and not inserted in the file NEWTAPE, if the NEW option has been specified previously.

VOIDT n

- a. The syntax is the same as the dollar sign VOID card.
- b. This option enables one to void an area on tape according to the range specified. Cards in the patch deck which lie in the voided range are unaffected and inserted.

ALGOL \$\$ CARD.

The \$\$ card is used to call out symbolic subprograms from a source language library file (made by the MAKCAST/DISK Program). The (ALGOL) file called may be a symbolic program to be compiled in its entirety, or it may be used as a patch or recipient of a patch. The format of this card is free form except for the following conditions:

- a. The two \$ symbols must appear in columns 1 and 2.
- b. Column 72 is blank.
- c. For proper sequence recognition of the source file by the ALGOL Compiler, columns 73-80 should contain zeros if the card is placed at the beginning of the source deck; or, if placed within the deck at the point of patching, the sequence field should contain the sequence number of the source program card previous to that of the first card image of the symbolic subprogram considered. The ALGOL Compiler does not overwrite that card image but only uses its sequence number as a reference point for patching.

The syntax of the \$\$ card is as follows:

$\langle \$\$card \rangle ::= \langle \text{symbolic tape id.} \rangle \langle \text{subprogram id.} \rangle \langle \text{partial program option} \rangle \langle \text{sequence no.} \rangle$

$\langle \text{symbolic tape id.} \rangle ::= A \mid B \mid C$

$\langle \text{partial program option} \rangle ::= [\langle \text{starting card number} \rangle : \langle \text{number of cards} \rangle] \mid \langle \text{empty} \rangle$

NOTE

The starting card number is the relative location of the card image in the subprogram, not the actual card sequence number.

The insertions are similar to those made when patching a OCRDIMG tape file with a card deck. The following example inserts the first eight cards of the CASTA subfile called FIEL into the program currently being compiled.

Example:

\$\$ A FIEL [1:8] $\langle \text{sequence number} \rangle$

COBOL \$\$ CARD.

The COBOL \$\$ card has the following format:

<u>Column</u>	<u>Contents</u>
1-2	\$\$
3	Blank
4-10	Subfile ID

The subfile ID in the above format refers to the subfile ID on the CASTA library file, and it is limited to a maximum of seven characters. The \$\$ card must appear before any other cards in the deck, except that \$ cards with a \$ sign in column 1 must precede the \$\$ card. Only one \$\$ card is allowed in a deck.

Cards following the \$\$ card are merged with the records in the CASTA subfile under sequence number control. The CASTA subfile may not contain the COPY FROM LIBRARY statement. The entire CASTA subfile is processed through the compiler. All \$ card options are permissible

except TAPE. Once the CASTA subfile has been processed, the SOLT file may be initiated with a \$ TAPE card.

SOURCE PROGRAM CARDS.

The discussion of these cards is covered separately for ALGOL and COBOL programs.

ALGOL SOURCE PROGRAMS.

Unless the source program is coming from tape, the last statement is END. The period in END. should not occur in card column 72.

Additional \$ cards are placed within the source program deck. The action taken is that specified by the last \$ card encountered during processing.

COBOL SOURCE PROGRAMS.

Unless the source program is coming from tape, the last statement is END-OF-JOB.

NINES CARD.

This card is used only if the source program is coming from tape and the last statement of the source program updating cards is not END. (for ALGOL) or END OF JOB. (for COBOL). The format of this card is:

<u>Column</u>	<u>Contents</u>
1-6	All 9's
7-72	Blanks
73-80	All 9's

FORTRAN TRANSLATOR CONTROL CARDS.

The FORTRAN Translator requires three control cards at all times and, in some cases, more. These Translator control cards are punched in the same format as the FORTRAN statements themselves. They are punched in columns 7-72 (blanks are ignored), and they may be continued on succeeding cards, providing other than a blank is punched in column 6 of each continuation card.

Each FORTRAN deck is preceded by two FORTRAN control cards. The first card should contain either TWO\$ or FOUR\$, depending on the language in which the program is written. The second is the START\$ card which will be discussed later.

The FORTRAN deck should be re-ordered so that each subprogram is placed ahead of all subprograms that reference it. Otherwise, forward procedure declarations will have to be inserted either in the ALGOL deck prior to compilation, or in the FORTRAN deck by use of A cards.

The final control card following the FORTRAN deck is the LAST\$ card. If the translation is for syntax only or if the ALGOL source deck is to be punched out, the FORTRAN programs may be stacked. A FINISH\$ card is then needed to end each job except the final one. A START\$ card must be included in each job. Only one TWO\$ or FOUR\$ card should be used for the entire stack.

The START\$ card contains all information needed to translate the FORTRAN program. The syntax for the START\$ card follows:

```

<START$ CARD> ::= START$ <empty> | START$ <start list> $
<start list> ::= <start list element> | <start list element>/
               <start list>
<start list element> ::= <punch options> | <FORTRAN listing option> |
                        <ALGOL listing option> | <tape file
                        request> | <sense switch option> | <sense
                        light option> | <ignore equivalence
                        option> | <ignore common option> | <global
                        options> | <octal option> | <syntax only
                        option> | <inhibit subscript switching
                        option> | <eliminate prefix option> | <disk
                        file option>
<punch options> ::= PC|PCO
<FORTRAN listing option> ::= SFL
<ALGOL listing option> ::= SAL
<tape file request> ::= TAPES, <tape list> | TAPES ←
                    <unsigned integer>
<tape list> ::= <unsigned integer> | <unsigned integer> ,
               <tape list>

```

```

<sense switch option> ::= SW, <sense list> | SW ←
                          <unsigned integer>
<sense light option> ::= SL, <sense list> | SL ←
                          <unsigned integer>
<sense list> ::= <unsigned integer> | <unsigned integer> ,
                  <sense list>
<ignore equivalence option> ::= XE
<ignore common option> ::= XC
<global options> ::= GLOBAL | GLODCL
<octal option> ::= OCTAL ← integer | OCTAL
<syntax only option> ::= SYNTAX
<inhibit subscript switching option> ::= ISS
<eliminate prefix option> ::= NOPFX
<disk file option> ::= DISK <disk description>
<disk description> ::= (<unit number>, <disk configuration>) |
                      (<unit number>, <disk configuration>),
                      <disk description>
<unit number> ::= <integer>
<disk configuration> ::= <disk item> | <disk item>,
                        <disk configuration>
<disk item> ::= <file type> | <disk access technique> |
               <number of areas> | <size of areas> |
               <size of logical record> | <size of physical
               record> <associated variable> | <save factor>
<file type> ::= PERM | TEMP
<disk access technique> ::= SERIAL | RANDOM | UPDATE
<number of areas> ::= AREAS ← <integer>
<size of areas> ::= LOGICAL RECORDS ← <integer>
<size of logical record> ::= WORDS ← <integer>
<size of physical record> ::= BLOCKED RECORD SIZE ← <integer>
<associated variable> ::= AV ← <integer variable>
<save factor> ::= SAVE ← <integer>

```

If <file type> is omitted, TEMP is assumed. If <disk access technique> is omitted, RANDOM is assumed. If <number of areas> is omitted, one area is assumed. <size of areas> is necessary for

all files that are being created. \langle size of logical record \rangle is necessary for all files. \langle size of physical record \rangle is necessary for blocked files and must be a multiple of the \langle size of logical record \rangle . \langle associated variable \rangle is necessary for all random files. \langle save factor \rangle is necessary for all permanent files that are being created.

Example:

```
START$ DISK (4, SERIAL, PERM, AREAS ← 2,  
LOGICAL RECORDS ← 1200, WORDS ← 10, BLOCKED RECORD SIZE ←  
150, AV ← INDEX, SAVE ← 10) $
```

would be translated as:

```
SAVE FILE DISK4 DISK SERIAL [2:1200] (2,10,150, SAVE 10);
```

This file will have the name 0000000/DISK4 on the disk file.

The \langle start list elements \rangle may be in any order and their effects are as follows:

- a. PC - an ALGOL deck is punched.
- b. PCO - an ALGOL deck is only punched. The ALGOL source program is not retained as a tape or disk file.
- c. SFL - suppress FORTRAN listing.
- d. SAL - suppress ALGOL listing.
- e. TAPES, n, m, etc. - FILE TAPE n 2(2,15);
FILE TAPE m 2(2,15); etc. are
declared.

TAPES ← n - the first n tapes are declared. If the tapes option is not present, the first 16 tapes are declared.

- f. SW, or SW← - sense switch statements are translated. A Boolean array SENSW [0:6] is set up. If SW, is used, all sense switches specified in <sense list> are set TRUE. All switches not specified are assumed FALSE. If SW← n is used, all switches through n are set FALSE. If the SW option is not used, the use of sense switch statements will cause a syntax error.
- g. SL, or SL← - sense light statements are translated. A Boolean array SENSL [0:4] is set up. If SL is used, all sense lights specified in <sense list> are set TRUE.
- All lights not specified are assumed FALSE. If SL← n is used, all lights through n are set FALSE. If the SL option is not used, the use of sense light statements will cause a syntax error.
- h. XE - all EQUIVALENCE statements which occur in the FORTRAN program are ignored.
- i. XC - all COMMON statements which occur in the FORTRAN program are ignored.
- j. GLOBAL - all identifiers in COMMON will appear as comments and will not be declared in the ALGOL program.
- k. GLODCL - All identifiers contained in COMMON in the first subprogram will be declared immediately before the first subprogram heading as individual items. COMMON will then be ignored in that subprogram, all succeeding subprograms, and the main program. This option should only be used if:

- 1) COMMON is present and identical in all subprograms and the main program.
 - 2) An identifier contained in COMMON does not appear in an EQUIVALENCE statement.
1. OCTAL - The Write Tape instructions are written in octal. The integer represents the record block size and cannot exceed 1023. If no integer is present, a record block of 256 is assumed.
 - m. SYNTAX - The ALGOL source program file is not retained and the ALGOL listing is suppressed.
 - n. ISS - The subscripts of the arrays are not reversed in the translation from FORTRAN to ALGOL. This option does not apply to arrays referenced by EQUIVALENCE or to an array referenced by COMMON unless the GLODCL is used. If the ISS option is not used, the array subscripts are automatically reversed.
 - o. NOPFX - The prefixes emitted by the FORTRAN Translator before all identifier names in the ALGOL translation are omitted. Identifier names that correspond to ALGOL reserved words or to identifiers used by the Translator are suffixed with the appropriate number of Q's to force the identifier name to contain seven characters. This is done to prevent duplication of identifiers.

In addition to the required control cards already discussed, one other control card is sometimes used. This card immediately precedes the first required control card and may be one of the following forms:

\$ CARD

\$ CARD NEW TAPE

```

$ CARD NEW TAPE RESEQ n
$ TAPE
$ TAPE NEW TAPE
$ TAPE NEW TAPE RESEQ n

```

where n represents an unsigned decimal number between 1 and 100,000. The \$ must be in column 1, and only one space should separate NEW and TAPE.

CARD indicates the FORTRAN source program is coming from the card input file CARD. TAPE indicates the FORTRAN source program is coming from the tape input file TAPE. When the FORTRAN source program is on file TAPE, the file CARD can be used to read in patch cards to the program. If patching is to be done, the FORTRAN source program must contain sequence numbers, strings of eight consecutive digits, in columns 73-80.

NEW TAPE indicates that a new copy of the FORTRAN source program is to be made which will include the patches, if any are present. RESEQ n will resequence the new copy in intervals indicated by the letter n.

If a \$ CARD is not present, the \$ CARD is assumed.

DECK STRUCTURE.

Following is an example of a deck set up to translate a FORTRAN source program from punched cards to an ALGOL source program which is stored on a tape labeled OCRDIMG. The ALGOL program is then compiled and executed.

```

? EXECUTE FORTRAN/TRANS
? DATA CARD
$ CARD

```

{ FORTRAN deck with
control cards }

? COMPILE ID1/ID2 ALGOL
? DATA CARD
\$ TAPE LIST

{ ALGOL patches
if any }

99999999
col. 73-80

? DATA (File Name)

Data

? END

The internal files of the FORTRAN Translator may be on disk.
These files are named as follows:

CARD - Card Input File (FORTRAN source)
LINE - Print Output File (FORTRAN & ALGOL source)
FORFIL - Tape Output File (ALGOL source)
PNCH - Punch Output File (ALGOL source)
TAPE - Tape Input File (FORTRAN source)
NEWTAPE - Tape Output File (FORTRAN source)

The files that are to be placed on disk require the use of a FILE
CARD immediately following the EXECUTE CARD. The FILE CARDS are:

? FILE CARD - ID1/ID2 DISK SERIAL
? FILE LINE - ID1/ID2 DISK SERIAL
? FILE FORFIL - ID1/ID2 DISK SERIAL
? FILE PNCH - ID1/ID/2 DISK SERIAL
? FILE TAPE - ID1/ID2 DISK SERIAL
? FILE NEWTAPE - ID1/ID2 DISK SERIAL

If the ALGOL source file FORFIL is placed on disk, an ALGOL FILE
card is needed immediately following the COMPILE card. The disk
file names of FORFIL and TAPE must be identical.

? ALGOL FILE TAPE - ID1/ID2 DISK SERIAL

When compiling the FORTRAN Translator, the following FILE cards must always be used:

? FILE LINE - LINE PRINT OR BACK UP TAPE
? FILE FORFIL - "OCRDMG" TAPE
? FILE PNCH - PNCH PUNCH
? FILE NEWTAPE - NEWTAPE

FORTRAN translator control cards are punched in the same format as the FORTRAN statements themselves. They are punched in columns 7-72, blanks are ignored, and they are continued on succeeding cards, providing a punch other than a blank is punched in column 6 of each continuation card.

Each FORTRAN deck is preceded by the two FORTRAN control cards. The first card should contain either TWO\$ or FOUR\$, depending on the language the program is written in. The second is a START\$ card.

The FORTRAN deck is re-ordered so that each subroutine is placed ahead of all subroutines which call upon it. Otherwise, forward procedure declarations will have to be inserted in the ALGOL deck prior to compilation.

Each FORTRAN deck should be followed by a LAST\$ card.

SYNTAX. The syntax for the START card is as follows:

$\langle \text{START\$ CARD} \rangle ::= \text{START\$} \langle \text{empty} \rangle \mid \text{START\$} \langle \text{start list} \rangle \$$
 $\langle \text{start list} \rangle ::= \langle \text{start list element} \rangle \mid \langle \text{start list element} \rangle / \langle \text{start list} \rangle$
 $\langle \text{start list element} \rangle ::= \langle \text{punch options} \rangle \mid \langle \text{FORTRAN listing option} \rangle \mid \langle \text{ALGOL listing option} \rangle \mid \langle \text{tape file request} \rangle \mid \langle \text{sense switch option} \rangle \mid \langle \text{sense light option} \rangle \mid \langle \text{suppress equivalence option} \rangle \mid \langle \text{suppress common option} \rangle \mid$

<global option> | <octal option> | <syntax
only option> | <inhibit subscript switching
option> | <eliminate prefix option> |
<disk file option>

<punch options> ::= PC|PCO

<FORTRAN listing options> ::= SFL

<ALGOL listing options> ::= SAL

SECTION 5 UTILITY ROUTINES

GENERAL.

There are a number of routines in the B 5500 programming system that are designed to facilitate the operation of the system. These routines are:

- a. Scheduling from disk.
- b. Symbolic library maintenance.
- c. Log maintenance.
- d. Disk directory.
- e. Printer backup.

For purposes of presentation, these routines have been arbitrarily called utility routines. The manner in which each routine performs its particular function will be discussed in detail in this section.

SCHEDULING FROM DISK.

The system program LDCNTRL/DISK and special features of the MCP provide a means whereby card deck information, including control information, can be placed on disk in the form of a pseudo card deck, and then used as though it were in a card reader. The LDCNTRL/DISK Program, pseudo card readers, and pseudo card decks are described in the following paragraphs.

LDCNTRL/DISK PROGRAM.

The system program LDCNTRL/DISK is a specially coded program which is partly contained within the MCP and partly contained as an object program on the system tape. The LDCNTRL/DISK Program can place either a magnetic tape file or a card file on the disk, and copy a file onto magnetic tape.

LOADING A CONTROL DECK FILE ONTO DISK. The primary function of the program LDCNTRL/DISK is to read a file with the \langle multiple file identification \rangle CONTROL and the \langle file identification \rangle DECK, and to place that file on disk in a special format, as one or more pseudo card decks. The file labeled CONTROL DECK may be a file in a card reader or a file on magnetic tape.

CARD READER CONTROL DECK FILE. If a CONTROL DECK file is to be read from a card reader, the file must be preceded by a LABEL CARD to identify it. Also, the last card in the CONTROL DECK must be an END CONTROL card, containing the information: ? END CONTROL.

MAGNETIC TAPE CONTROL DECK FILE. If a CONTROL DECK file is to be copied from magnetic tape onto disk, the tape must be properly labeled and, as is the case with a CONTROL DECK from a card reader, the last card image on the tape file must be an ? END CONTROL card. In addition to these requirements, the tape file must be properly formatted so that question mark cards (i.e., control card and program-parameter cards) can be recognized. Specifically, the tape must have the following characteristics:

- a. The tape must be unblocked.
- b. Each record containing a question mark card is recognized as 9 words in length.
- c. Each record containing a card which is not a question mark is recognized as 10 words in length.

PSEUDO DECKS ON DISK. When the LDCNTRL/DISK Program reads a CONTROL DECK file, it places it on disk as one or more pseudo card decks. The number of pseudo decks created depends upon the number of ? END cards located within the CONTROL DECK. That is, each time a ? END card is encountered, it is taken to denote the end of a deck; creation of another pseudo deck is then initiated. As each new pseudo deck is created, it is given an identification of the form: # <integer>.

It should be noted that what is referred to as a pseudo deck is analogous to a single continuous deck that would be placed in a card reader. Therefore, if a pseudo deck contains more than one file, each file following the first will be recognized only when the file preceding it has been passed.

It should also be noted that there is no set limit to the number of cards that may be contained in a CONTROL DECK file, but a pseudo card deck (the end of which is denoted by a ? END card) can contain no more than 12,000 cards.

REMOVING PSEUDO DECKS FROM DISK. When each pseudo card deck is placed on disk, the deck is linked to the previous deck, forming a queue waiting to be used by a pseudo card reader. Because of the queue feature, the RD keyboard input message must be used to remove pseudo decks from the disk.

COPYING A CONTROL DECK ONTO TAPE. The secondary function of the LDCNTRL/DISK Program is to read a file labeled CONTROL DECK, delimited by a ? END CONTROL card, and to copy it onto magnetic tape. If the CONTROL DECK being copied is a card file, the file will be copied onto tape in the required format specified above (see page 5-2). If the CONTROL DECK being copied is a magnetic tape file, a tape copy is performed.

CALLING THE LDCNTRL/DISK PROGRAM OUT FOR EXECUTION. The LDCNTRL/DISK may be called out either by a keyboard input message or control cards.

If LDCNTRL/DISK is to be executed to place a CONTROL DECK on the disk, the keyboard input message

LD DK

may be used or a control card containing

? EXECUTE LDCNTRL/DISK

may be used.

If LDCNTRL/DISK is to be executed to copy a CONTROL DECK onto tape, the keyboard input message

LD MT

may be used on control cards containing

```
? EXECUTE LDCNTRL/DISK
? COMMON = 1
```

may be used.

PARITY ON A CONTROL DECK MAGNETIC TAPE FILE. If a parity error is encountered in a CONTROL DECK file being read from magnetic tape, the parity file is skipped. In effect, the file containing parity is completely ignored.

PSEUDO CARD READERS AND THE USE OF PSEUDO CARD DECKS.

To make use of pseudo card decks, the MCP contains logic which can, in effect, supply the system with up to 32 pseudo card readers. These pseudo card readers in many ways appear to be much like physical peripheral units. That is, system messages are typed for the pseudo card readers as though they were card readers, and keyboard input messages can reference the pseudo card readers. The pseudo card readers are identified by the <unit mnemonic>s:

CDA through CDZ, excluding CDI
and CDO, and CD2 through CD9.

At HALT-LOAD time, all pseudo card readers are turned off. The system operator may cause these readers to be turned on through use of an RN keyboard input message.

THE RN MESSAGE TO TURN ON PSEUDO CARD READERS. When an RN <digit> message is initially entered and the <digit> is not equal to zero, the MCP automatically searches for pseudo card decks to satisfy the need of the specified number of pseudo card readers. Thereafter, as long as pseudo card readers are on and pseudo card decks are available, the MCP will keep the readers loaded.

THE RN MESSAGE TO TURN OFF PSEUDO CARD READERS. If the system operator wishes to turn off pseudo card readers, he need only type in an RN message that specifies the number of pseudo card readers he wants left on. The MCP will then turn off a sufficient number of readers to meet these requirements as soon as the readers complete processing their current deck.

REMOVING DECKS FROM PSEUDO CARD READERS. If, for any reason, it is desired to remove a deck from a pseudo card reader (e.g., a card file never opened by a program that was discontinued), the removal can be accomplished by entering an ED keyboard input message.

HANDLING OF CONTROL CARD ERRORS IN PSEUDO CARD DECKS. If, while a pseudo card deck is being read, an error is detected in a control card or program-parameter card, the MCP will remove the deck in which the erroneous card appears and will continue to the next available pseudo deck. Load, DUMP, or REMOVE operations are not permitted on pseudo decks.

SYMBOLIC LIBRARY FILE ON DISK.

The symbolic library facility (MAKCAST/DISK) is such that library files can exist on disk as well as on magnetic tape. In making this disk capability available, the ALGOL Compiler file CASTA and the COBOL file LIBRAR have been set up so that the library files they reference are expected to be on disk. (The ALGOL Compiler files CASTB and CASTC are set up to expect library files on magnetic tape.)

Although the standard media for symbolic library files is as noted above, the media can be specified through use of label equation cards. Also, if it is desired to change the standard setup for the files, that change can be accomplished by changing the file declarations for those files, and recompiling the compilers.

If label equation cards are to be used in reference to the files for symbolic libraries, or if file declarations are to be modified, the makeup of the file declarations within the compilers must be known.

The card images for the symbolic library files in the ALGOL Compiler contain the following information (BUFFSIZE is DEFINED equal to 56.):

File Declaration

```
FILE CASTA DISK SERIAL "CASTA" "LIBRARY" (1, BUFFSIZE);
FILE CASTB (1, BUFFSIZE);
FILE CASTC (1, BUFFSIZE);
```

The card images for the symbolic library file in the COBOL Compiler contain the following information:

File Declaration

```
FILE LIBRAR DISK SERIAL "CASTA" "LIBRARY" (1, 56);
```

An example of a label equation card which could be used to specify disk as the media for an ALGOL symbolic library file is as follows:

```
? ALGOL FILE CASTC = CASTC/LIBRARY SERIAL
```

An example of a label equation card which could be used to specify tape as the media for a COBOL symbolic file is as follows:

```
? COBOL FILE LIBRAR = CASTA TAPE
```

CONTROL CARD SYNTAX.

The expanded control card syntax is as follows:

```
<control card> ::= $$$ <master control card> | <code>
                  <subcontrol card> | $$$ <end card>
<code> ::= $$$ | <continuation code>
<continuation code> ::= $-$
<master control card> ::= DISPLAY <tape identifier> <master display
                           features> | MAKE <tape identifier> <master
                           make features> | MAKE <tape identifier>
                           FROM <tape identifier> <master make
                           features>
<tape identifier> ::= A|B|C
```

```

<master display features> ::= LIST | PUNCH | DIR | SINGLE | <master
display features> LIST | <master
display features> PUNCH | <master
display features> DIR | <master display
features> SINGLE | <empty>
<master make features> ::= <master display features> | COPY |
<master make features> COPY | <empty>
<subcontrol card> ::= <display types> | <make types>
<end card> ::= END
<display types> ::= <subprogram identifier> <list-punch option>
<list-punch option> ::= LIST | PUNCH | <list-punch option> LIST |
<list-punch option> PUNCH | <empty>
<make types> ::= DELETE <subprogram identifier> <list-punch option> |
REPLACE <subprogram identifier> <replacement option>
<rename option> <make features> | PATCHA
<subprogram identifier> <location option>
<rename option> <make features> |
PATCHC <subprogram identifier> <location option>
<rename option> <make features> | ADD <subprogram
identifier> <location option> <insertion option>
<rename option> <make features> | <subprogram
identifier> <location option> <rename option>
<make features>
<subprogram identifier> ::= <identifier>
<replacement option> ::= WITH <subprogram identifier> <location
option> | <empty>
<rename option> ::= RENAME <subprogram identifier> | <empty>
<location option> ::= ON <tape identifier> | ON OCRD | <empty>
<make features> ::= <list-punch option> | <sequence option> |
<make features> <list-punch option> | <make
features> <sequence option> | <empty>
<sequence option> ::= SEQA <increment> | SEQC <increment> | <empty>
<increment> ::= 10|100|1000|10000|100000|1000000|10000000
<insertion option> ::= AFTER <subprogram identifier> | END | <empty>

```


SEMANTICS. The control cards for the symbolic library maintenance routine are designed to provide the features needed in creating, updating, and displaying a symbolic library.

There are three kinds of control cards: Master, Subcontrol, and End cards. The first three columns of the Master and End cards must contain the control card flag: \$\$\$\$. The first three columns of the Subcontrol card may contain \$\$\$ or the continuation code: \$-\$. The continuation code is provided in case one Subcontrol card is not enough to contain all of the required subcontrol options. As many Continuation cards as necessary may be used, with all cards except the last one in the group starting with \$-; the last Continuation card in the group must have \$\$\$.

The information on all control cards is free-field, with the restriction that each syntactically defined element must be separated from the next by at least one blank column. Further, a syntactically defined element may not be split across two Continuation cards. Normally, all control-card information must be entered in columns 1-72, with columns 73-80 reserved for comments. However, a percent sign (%) inserted anywhere on a control card (except columns 1-3, which must contain the control card flag) will have the effect of marking the end of the card; comments may be inserted after the percent sign.

The Master control card indicates whether a new library file is to be made (MAKE) or information about an existing library is required (DISPLAY). If a new library is being made, the Master control card indicates whether an existing library is to serve as a base for the new library (MAKE FROM). If FROM is not used, the new library will contain only those subprograms called for explicitly in Subcontrol cards. When FROM is used, all subprograms on the existing input library will be transferred to the new library unless Subcontrol cards indicate otherwise (e.g., REPLACE or DELETE). If a library is to be displayed (DISPLAY), the Master control card specifies the input source: CASTA, CASTB, or CASTC.

Other reserved words appearing in the control cards have the following meanings:

- a. DIR specifies that a directory of all subprograms on the library is to be printed. (For MAKE functions, the directory reflects the contents of the new output library; for DISPLAY functions, the directory reflects the contents of the input library.) The number of subprograms that can be recorded on one library is limited by the directory size; the directory will hold up to 1344 characters. Each identifier to be placed in the directory, corresponding to a subprogram placed on the library, will use $N + 4$ characters, where N is the number of characters in the subprogram identifier.
- b. A, B, and C refer to the three possible subprogram libraries which can be referenced. The libraries which are labeled CASTA, CASTB, and CASTC are referenced in the control cards as A, B, and C, respectively.
- c. OCRD refers to a tape labeled OCRDIMG being used as input.
- d. PATCHA and PATCHC are used to distinguish patching of ALGOL and COBOL subprograms. PATCHA (ALGOL patching) utilizes sequence numbers as they appear in columns 73-80 of the data cards. PATCHC (COBOL patching) utilizes sequence numbers as they appear in columns 1-6 of the data cards.
- e. SEQA and SEQC are used to distinguish resequencing of ALGOL and COBOL subprograms. SEQA 100 (ALGOL resequencing), for example, specifies that the data cards are to be resequenced in columns 73-80, with the resulting cards numbered 00000100, 00000200, 00000300, ..., etc. Similarly, SEQC 100 (COBOL resequencing) specifies that the data cards are to be resequenced in columns 1-6, with the resulting cards numbered 001000, 002000, 003000, ..., etc.

- f. LIST, PUNCH, or both, may appear on the Master and/or the Subcontrol cards. When LIST and/or PUNCH appear on the Master control card, the effect is that of having requested the indicated option(s) on every Subcontrol card which follows.
- g. SINGLE appearing on a Master control card specifies that page skipping is not to be performed on the line printer output. When SINGLE is omitted, the listing for each subprogram will begin at the top of a new page; also, each control card will be printed at the top of a new page.
- h. COPY appearing on a MAKE or MAKE FROM Master control card specifies that an additional copy of the new library is to be made on tape.

MAINTENANCE FUNCTION EXAMPLES. The following are maintenance function examples.

- a. DISPLAY function.

Example:

```

$$$ DISPLAY A DIR
$$$ END

```

- 1) This control deck would yield the directory of a CASTA file.

Example:

```

$$$ DISPLAY A LIST DIR
$$$ END

```

- 2) When no Subcontrol cards are present to select specified subprograms, the LIST-PUNCH option on the Master control card will cause all subprograms on the file to be listed and/or punched, as requested. The above example would yield a directory of the CASTA library, and a listing of all subprograms on the library.

Example:

```
$$$ DISPLAY A PUNCH DIR
$$$ SUB1
$$$ SUB2 LIST
$$$ SUB3
$$$ END
```

- 3) This control deck would yield a punched card output of SUB1, SUB2, and SUB3, a listing of SUB2, and a directory of the CASTA library.
- 4) When subprograms are punched, the card output for each subprogram includes a header card; this card contains \$\$\$ in columns 1-3 and the subprogram identifier starting in column 5.

b. MAKE function (ADD).

Example:

```
$$$ MAKE A FROM B DIR LIST
```

```
$$$ ADD SUB1 SEQA 10 PUNCH
```

(Followed by data cards, which cannot have \$\$\$ or \$-\$ in columns 1-3.)

```
$$$ ADD SUB2 SEQC 100
```

(Followed by data cards, which cannot have \$\$\$ or \$-\$ in columns 1-3.)

```
$$$ ADD SUB3 ON OCRD AFTER B4 PUNCH
```

```
$$$ ADD C4 ON C END
```

```
$$$ ADD C2 ON C RENAME NEW
```

```
$$$ END
```

- 1) The following shows the effect of using the above card deck.

<u>OCRDIMG Input Tape</u>	<u>Master Input Library B</u>	<u>Input Library C</u>	<u>New Library A</u>	<u>Other Output</u>
.	B1	C1	SUB1	Listing and card output
.	B2	C2	SUB2	Listing
.	B3	C3	B1	Listing
Card images	B4	C4	B2	Listing
corresponding	B5	C5	B3	Listing
to SUB3			B4	Listing
.			SUB3	Listing and card output
.			B5	Listing
.			C4	Listing
			NEW	Listing
				Directory of new CASTA library

- 2) The ADD function is allowed only in conjunction with the MAKE FROM function. (An ADD Subcontrol card cannot reference a subprogram on the master input library, e.g., the CASTB library in the above example.) When the ADD Subcontrol card does not locate the source of the subprogram (e.g., ADD XYZ ON A), it is assumed that the subprogram is in the card reader following the ADD Subcontrol card. Note that LIST appearing on the Master control card not only has the effect of having requested the LIST option on each of the subprograms referenced in Subcontrol cards, but additionally will cause a listing of each subprogram transferred from the master input library to the new library. The ADD -- END card specifies that the referenced subprogram is to follow any remaining subprograms on the master input library.

c. MAKE function (DELETE).

Example:

```
$$$ MAKE A FROM B LIST
$$$ ADD C5 ON C AFTER B3
$$$ DELETE B4
$$$ ADD C2 ON C PUNCH
$$$ DELETE B6 LIST
$$$ END
```

- 1) The following shows the effect of using the above card deck.

<u>Master Input</u> <u>Library B</u>	<u>Input</u> <u>Library C</u>	<u>New</u> <u>Library A</u>	<u>Other Output</u>
B1	C1	B1	Listing
B2	C2	B2	Listing
B3	C3	B3	Listing
B4	C4	C5	Listing
B5	C5	C2	Listing and card output
B6	C6	B5	Listing
B7			Listing (of B6)
B8		B7	Listing
		B8	Listing

- 2) The DELETE function is allowed only in conjunction with the MAKE FROM function. (A DELETE Subcontrol card can reference only a subprogram on the master input library, e.g., the CASTB library in the above example.) Note that subprograms to be deleted are not listed or punched when LIST or PUNCH appears on the Master control card. If listing and/or punching of a subprogram to be deleted is desired, then LIST and/or PUNCH must appear on the DELETE Subcontrol card.

REPLACE B4

- 4) To replace B4 with a subprogram in the reader which is to be renamed CARDS, the Subcontrol card would state:

REPLACE B4 WITH CARDS

or

REPLACE B4 RENAME CARDS

- 5) To replace B4 with the subprogram on a OCRDIMG tape which is to be renamed NEW, the Subcontrol card would state:

REPLACE B4 WITH NEW ON OCRD

- 6) To replace B4 with the subprogram on a OCRDIMG tape which is to have the same name, the Subcontrol card would state:

REPLACE B4 WITH B4 ON OCRD

- 7) A subprogram to be replaced cannot be listed or punched, if such action is desired, use the DELETE function (see page 5-13).

e. MAKE function (identifier).

Example:

```
$$$ MAKE B DIR
$$$ YXZ LIST
(Followed by data cards, which cannot have $$$ or
$-$ in columns 1-3.)
$$$ A1 ON A
$$$ B3 ON B LIST
$$$ C2 ON C
$$$ A2 ON A PUNCH LIST
$$$ END
```

d. MAKE function (REPLACE).

Example:

```
$$$ MAKE A FROM B
$$$ DELETE B2
$$$ ADD C3 ON C
$$$ REPLACE B4 WITH C1 ON C LIST
$$$ REPLACE B6 RENAME CARDPROGRAM LIST
(Followed by data cards which cannot have $$$ or
$-$ in columns 1-3.)
$$$ END
```

- 1) The following shows the effect of using the above card deck.

<u>Master Input</u> <u>Library B</u>	<u>Input</u> <u>Library C</u>	<u>New</u> <u>Library A</u>	<u>Other Output</u>
B1	C1	B1	
B2	C2	C3	
B3	C3	B3	
B4	C4	C1	Listing
B5	C5	B5	
B6	C6	CARDPROGRAM	Listing

- 2) The REPLACE function is allowed only in conjunction with the MAKE -- FROM function. (A REPLACE Subcontrol card can reference only a subprogram on the master input library, e.g., the CASTB library in the above example.) When the new subprogram to be added is in the card reader, the REPLACE Subcontrol card does not specify WITH -- ON. For subprograms located on an A, B, C, or OCRDIMG tape, the reserved words WITH -- ON will specify the source.
- 3) For example, to replace B4 with a subprogram in the reader which is to have the same name, the Subcontrol card would state:

- 1) The following shows the effect of using the above card deck.

<u>Input Library A</u>	<u>Input Library B</u>	<u>Input Library C</u>	<u>New Library B</u>	<u>Other Output</u>
A1	B1	C1	XYZ	Listing
A2	B2	C2	A1	
A3	B3	C3	B3	Listing
A4	B4	C4	C2	
	B5		A2	Listing and card
	B6			output Directory of CASTB library

- 2) A Subcontrol card starting with a subprogram identifier is allowed in conjunction with both the MAKE and MAKE -- FROM functions. This type of Subcontrol card, when used with the MAKE -- FROM function, cannot reference a subprogram on the master input library.

f. MAKE function (PATCH).

Example:

```

$$$ MAKE A FROM B
$$$ PATCH C B1 SEQC 100
(Followed by patch cards, which cannot have $$$ or
$-$ in columns 1-3.)
$-$ PATCHA XYZ ON
$$$ OCRD SEQA 10
(Followed by patch cards, which cannot have $$$ or
$-$ in columns 1-3.)
$$$ ADD CARDPROGRAM
(Followed by data cards, which cannot have $$$ or
$-$ in columns 1-3.)
$$$ END

```

- 1) The PATCH function is allowed in conjunction with both the MAKE and MAKE -- FROM functions. A PATCH Sub-control card may reference a subprogram on any subprogram tape or the OCRDIMG tape. If PATCH is used with the MAKE -- FROM function and the Subcontrol card does not specify ON (e.g., PATCHA A3), it will be assumed that the source program to be patched is on the master input library.
- 2) Patching will be performed according to the collating sequence. For example, a card sequenced as bbbbA100 will precede a card sequenced as bbbbB200. A card sequenced as bb340000 will precede a card sequenced as 00000010.
- 3) In regard to sequence number:
 - a) When the patch card is less than (<) the subprogram card, the patch card will be selected.
 - b) When the patch card is equal to (=) the subprogram card, the patch card will replace the subprogram card.
 - c) When the patch card is greater than (>) the subprogram card, the subprogram card will be selected.
- 4) The user will find it convenient at times to use a dummy call on the PATCH function. If, for example, we have a CASTB library with subprograms B1, B2, B3, . . . , B10, and we wish to generate a new CASTB library exactly the same except for resequencing of subprogram B5, the simplest control deck to accomplish such a function would be:

```

$$$$ MAKE B FROM B
$$$$ PATCHA B5 SEQA 100
$$$$ END

```

- 5) With no B5 patch cards in the reader, the effect will be simply to resequence B5. All the subprograms will retain their original order.

END OF JOB AND ERROR MESSAGES.

The line printer or printer-backup file is used to log error messages or the job-complete message. The following messages may appear during a run of MAKCAST:

- a. JOB COMPLETE. NO ERROR SITUATION ENCOUNTERED.
- b. JOB TERMINATED. ERROR IN MASTER CONTROL CARD.
- c. JOB TERMINATED. ERROR IN SUBCONTROL CARD.
- d. JOB TERMINATED. ITEM NOT FOUND IN DIRECTORY.
- e. JOB TERMINATED. DIRECTORY HAS IMPROPER FORMAT.
- f. JOB TERMINATED. REFERENCED PROGRAM IS NOT FORWARD ON THE MASTER LIBRARY.
- g. JOB TERMINATED. OUTPUT TAPE DIRECTORY OVERFLOW.

SETUP.

The setup to make a CAST library tape is as follows:

- a. ? EXECUTE MAKCAST/DISK.
- b. ? DATA CARD.
- c. Master control card.
- d. First Subcontrol card.
 - 1) First subprogram card images.
- e. Second Subcontrol card.
 - 1) Second subprogram card images.
- .
- .
- .
- .
- . Last Subcontrol card.
 - . Last subprogram card images.

- . End Card.
- . ? END.

COPYING SYMBOLIC LIBRARY TAPES ONTO DISK.

To change existing library files from magnetic tape onto disk, a program need only perform a direct copy onto disk. The copying program should use read symbolic library tapes as unblocked 56-word records and should write them on disk in the same fashion. Examples of file declarations which could be used by a copying program written in ALGOL are:

```
FILE IN CASTA (2, 56);  
FILE OUT CASTA DISK SERIAL [20:240] "CASTA" "LIBRARY"  
(2, 56, SAVE 30);
```

LOG MAINTENANCE.

Log information for programs run on the system is written in a file on user disk. The log file occupies one area on disk, and has the \langle multifile identification \rangle SYSTEM and the \langle file identification \rangle LOG. It is the user's responsibility to provide this file.*

The file SYSTEM/LOG is blocked. There are six logical records per physical record. The logical records are five words (i.e., 40 characters) in length; the physical records are 30 words in length.

LOG ENTRY SPECIFICATIONS.

Entries in the log can be considered to fall into one of three categories:

- a. Compile and go entries.
- b. Compile only entries.
- c. Execute entries.

With respect to these categories, the following rules determine how a program is entered in the log:

* For information as to how this area should be reserved, reference should be made to page 3-24.

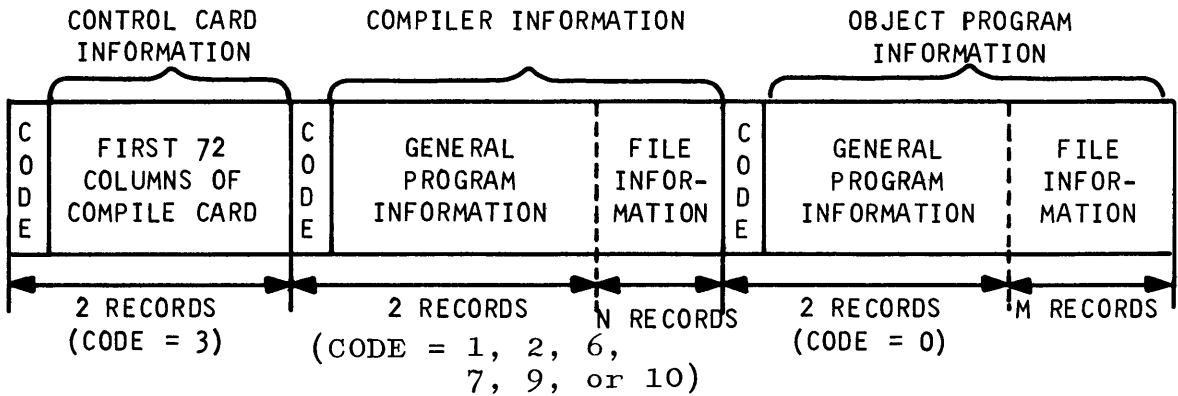
- a. If a compile-and-go is made and the program being compiled contains no syntax errors, the log information for both the compiler and the object program is listed in a compile-and-go entry.
- b. If a compile-and-go run is made and the program being compiled contains syntax errors, if a compile-for-syntax run is made, or if a compile-to-library run is made, the log information for the compiler is listed in a compile-only entry.
- c. If an execute run (i.e., library call out) is made, the log information for the object program is listed in an execute entry.

The general format of each of the three types of log entries is shown in figure 5-1. The first log entry starts in the record with relative address 1.

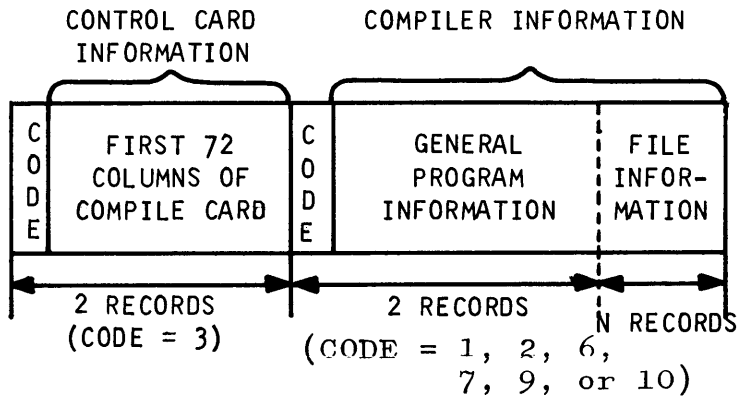
CODE WORD. As shown, each log entry contains (1) control card information and (2) compiler and/or object program information. The code word preceding each group of information denotes the type of information. That is, information preceded by a 1 pertains to the ALGOL Compiler; information preceded by a 2 pertains to the COBOL Compiler; information preceded by a 3 pertains to an object program. Code 4 denotes the end of log information, and code 5 pertains to printer back-up information. Since under SHAREDISK there is one system log for all systems, the system ID (0,1,2,3) is placed in the [1:2] field of the code word in each log entry.

CONTROL CARD INFORMATION. Control card information is contained in the first two records of a log entry, starting at the second word of the first record. This information is a copy of the contents of the first 72 columns of the COMPILE card or EXECUTE card that caused the particular run to be scheduled.

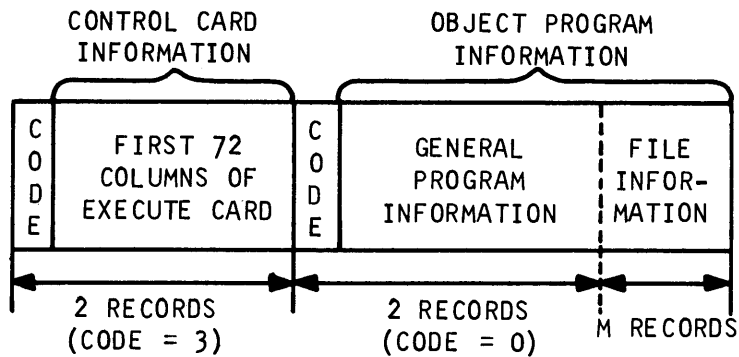
COMPILE AND GO ENTRY



COMPILE ONLY ENTRY



EXECUTE ENTRY



NOTE

N = Number of files declared by compiler.
M = Number of files declared by object program.
CODE (WORD) ::= <SYSTEM ID=[1:2]> <CODE=[3:45]>.

Figure 5-1. Log Entry Formats

The word immediately preceding control card information is a code with the integer value 3.

COMPILER AND OBJECT PROGRAM INFORMATION. Compiler information and object program information have identical formats; therefore, the format of this information is discussed under the general name, program information.

Program information falls into two categories: general information and file information. The general program information is contained in two records. The file information, which is a copy of the FPB (File Parameter Block) for the program, requires a variable number of records, depending on the number and use of files declared by the program. If a file is closed more than once or uses more than one physical reel of tape, an additional record appears in the log for each additional closing or tape reel.

The format of general program information in a log entry (including the code word) is shown in figure 5-2 and described below.

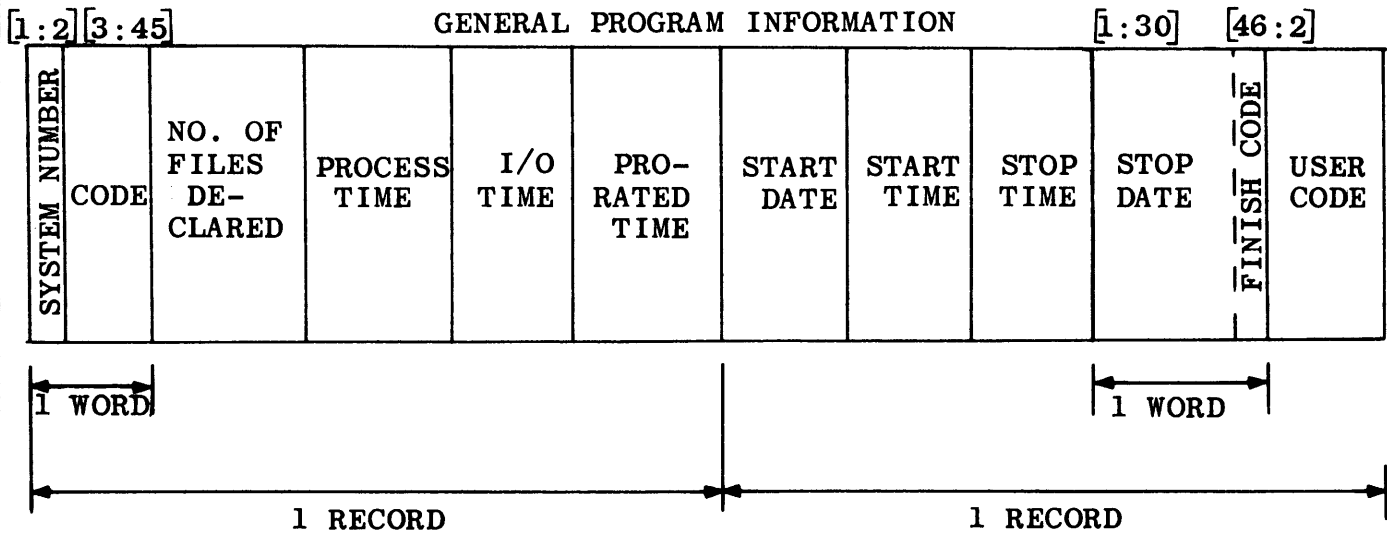


Figure 5-2. Format of General Program Information

<u>Entry</u>	<u>Description</u>
CODE[1:2]	INTEGER - 0, 1, 2, 3 for respective system A, B, C, D
CODE[3:45]	INTEGER - 0 - object program was executed 1 - ALGOL Compiler was executed 2 - COBOL Compiler was executed 3 - control card information 4 - end of log 5 - printer backup 6 - FORTRAN Compiler was executed 7 - BASIC Compiler was executed 8 - disk file 9 - XALGOL Compiler was executed 10 - TSPOL Compiler was executed
NO. OF FILES DECLARED	INTEGER
PROCESS TIME	INTEGER - time in 60ths of a second
I/O TIME	INTEGER - time in 60ths of a second
PRORATED TIME	INTEGER - time in 60ths of a second
START DATE	BCL - YYDDD format, e.g., 65046 (The YYDDD format provides that the YY characters specify the last two digits of the year, and the DDD characters specify the number of the day of the year in Julian format. The number is right justified.)
START TIME	INTEGER - time in 60ths of a second since Halt/Load time
STOP TIME	INTEGER - time in 60ths of a second since Halt/Load time

Entry

Description

STOP DATE .[1:30]

BINARY - YYDDD format, e.g., 65047 (Refer to START DATE.)

FINISH CODE .[46:2]

BINARY - 0 - end of job
1 - syntax error
2 - DSed or ESed
3 - abort

USERCODE

BCL - A₁A₂A₃A₄A₅A₆A₇

The format of one file-information record is shown in figure 5-3 and described below.

FILE INFORMATION

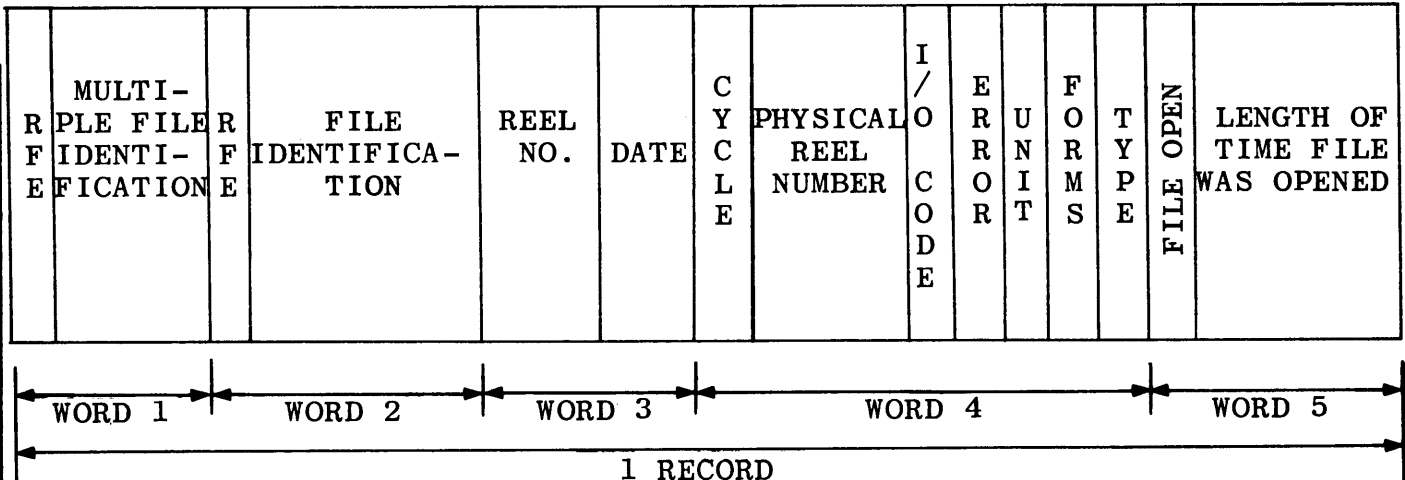


Figure 5-3. Format of One File-Information Record

<u>Word</u>	<u>Entry</u>	<u>Field</u>	<u>Contents</u>
1	MULTIPLE FILE IDENTIFICATION	[6:42]	7-character multifile identification
2	FILE IDENTIFICATION	[6:42]	7-character file identification
3	REEL NO.	[0:18]	Reel number (three alphabetic characters)
	DATE	[18:30]	Creation date in five characters (YYDDD)

<u>Word</u>	<u>Entry</u>	<u>Field</u>	<u>Contents</u>
4	CYCLE	[0:6]	Cycle number (binary)
	PHYSICAL REEL NUMBER	[6:17]	Physical reel number (binary)
	I/O CODE	[23:1]	Tape I/O code 0 - output 1 - input
	ERROR	[24:12]	Number of errors in handling this file (binary)
	UNIT	[36:6]	I/O unit used by this file (binary). This number cor- responds to Logical Unit Number plus one.

<u>Value</u>	<u>I/O Unit</u>
0	Not opened
1	MTA
2	MTB
3	MTC
4	MTD
5	MTE
6	MTF
7	MTH
8	MTJ
9	MTK
10	MTL
11	MTM
12	MTN
13	MTP
14	MTR
15	MTS
16	MTT
17	DRA
18	DRB
19	DKA
20	DKB
21	LPA
22	LPB
23	CPA
24	CRA

<u>Word</u>	<u>Entry</u>	<u>Field</u>	<u>Contents</u>	
			<u>Value</u>	<u>I/O Unit</u>
			25	CRB
			26	SPO
			27	PPA
			28	PRA
			29	PPB
			30	PRB
			31	DCA
	FORMS	[42:1]	1 indicates special forms required	
	TYPE	[43:5]	Type of file (binary)	
			<u>Value</u>	<u>Type</u>
			0	CP/CR
			1	LP only
			2	MT
			3	DG (designated)
			4	LP/PBT
			5	Specified unit (unlabeled)
			6	PBT only
			7	PT
			8	PT unlabeled
			9	MT unlabeled
			10	Disk random
			11	SPO
			12	Disk serial
			13	Disk update
			14	Data communications (input or output)
			15	PBD only
			16	PBT/PBD
			17	LP/PBD
			18	LP/PBT/PBD
			19	REMOTE - data communications (input/output)

<u>Word</u>	<u>Entry</u>	<u>Field</u>	<u>Contents</u>
5	FILE OPEN	[1:1]	1 indicates file open
	LENGTH OF TIME FILE WAS OPENED	[2:46]	I/O time on this unit in 60ths of a second (binary). Cumula- tive time from INITIATEIO to IOFINISH

SPECIAL RECORDS AND LOG INITIALIZATION.

RECORD ZERO. The first record in SYSTEM/LOG (i.e., the record with relative address 0) is used by the MCP when making log entries. The value of the first word in record zero specifies the number of records written in the log. The value of the second word specifies the record capacity of the log. The third and fourth words are used in conjunction with the warning messages supplied by the MCP which signify when the log is half-full and full. The fifth word contains, in BCL, DISKLOG.

RECORD n + 1. The first word of the record immediately following the last log entry contains a code with the value 4. This record denotes the end of log information, and it is not included in the value contained in the first word record of record zero.

INITIALIZING THE LOG. If a user program wishes to initialize the log (i.e., set up the log so that the MCP considers the log empty), the following action must be performed:

- a. The 1st, 3rd, and 4th words in record zero must be set to zero.
- b. The 1st word in record 1 must be set to 4.

REMOTE LOG SPECIFICATIONS. The remote log information for the data communications facilities is written in a file on the user disk. The file has the <file identification prefix> "REMOTE" and the <file identification> "LOG." The file REMOTE/LOG is blocked and must be confined to one area on the disk. There are five logical records per physical record. A logical record is five words in length or forty characters; a physical record is thirty words in length. It is the user's responsibility to provide this file. Logging for data communications is bypassed if the system does not provide a REMOTE/LOG file.

LOG ENTRY SPECIFICATIONS. Entries in the Remote Log can be considered to be of five types:

- | | |
|--------|---|
| Type 1 | Log-Out Entry |
| Type 2 | Log-In Entry |
| Type 3 | Control Card Entry of less than 32 characters |
| Type 4 | Control Card Entry of 32 characters or more, not greater than 72 characters |
| Type 5 | Job Statistics Entry |

Type 1, Type 2, and Type 3 entries each require one logical record in the log. Types 4 and 5 require two logical records per entry.

Type 1 LOG-OUT Entry. The following information is entered into the file REMOTE/LOG when a data communications station logs out.

Word 0 [9:9] Station Number ([9:4]=TU,[14:4]=BUF)
[42:6] Code = 1

Word 1 User Identification (as specified by the
FILE SECURITY SYSTEM)

1 Record

Word 2 Current Date (YYDDD-BCL)

Word 3 Time of day at Log-Out

Word 4 Unused

Type 2 LOG-IN Entry. The MCP enters the following information in the file REMOTE/LOG when a data communications station logs in.

Word 0 [9:9] Station Number ([9:4]=TU,[14:4]=BUF)
[42:6] Code = 2

Word 1 User Identification (as specified by the
FILE SECURITY SYSTEM)

1 Record

Word 2 Current Date (YYDDD-BCL)

Word 3 Time of day at Log-In

Word 4 Unused

Type 3 CONTROL CARD Entry (31 characters or less). The MCP enters the following information -- or Type 4 information -- in the file REMOTE/LOG when a job is selected to run. Every RUN or EXECUTE from a remote station is logged.

Word 0 [9:9] Station Number ([9:4]=TU,[14:4]=BUF)
 [18:24] RUN NUMBER*
 [42:6] Code = 3

1 Record

Word 1
 thru Contents of Control Card
 Word 4

Type 4 CONTROL CARD Entry (32 characters up to 72 characters). The MCP enters the following information -- or Type 3 information -- in the file REMOTE/LOG when a job is selected to run. Every RUN or EXECUTE from a remote station is logged.

Word 0 [9:9] Station Number ([9:4]=TU,[14:4]=BUF)
 [18:24] RUN NUMBER*
 [42:6] Code = 4

2 Records

Word 1
 thru Contents of Control Card
 Word 9

Type 5 JOB STATISTICS. The MCP enters the following information in the file REMOTE/LOG when a station detaches from a job.

Word 0 [2:1] 1 if this station attached by entering an EXECUTE or RUN card; 0 if attached by READ SEEK or WRITE

[9:9] Station Number

[18:24] RUN NUMBER (as specified in the Type 3 or Type 4 Entry)

[42:6] Code = 5

Word 1 User Code

* Entries in the file REMOTE/LOG corresponding to entries in the file SYSTEM/LOG have the same RUN NUMBER, where a job's RUN NUMBER is defined to be its start time -- in the 60ths of a second -- as specified in the System Log.

Word 2		First name of the object program (7 characters)
Word 3		Second name of the object program (7 characters)
Word 4		Processor Time in 60th of a second (i.e., processor time used for this station, out of total used by job)
Word 5		Pro-Rated Time in 60th of a second (i.e., pro-rated time used by this station, out of total used by job)
Word 6		I/O Time in 60th of a second (i.e., I/O time used by this station, out of total used by job)
Word 7	[3:21]	Start Date--Date when job attached to this station (in binary)
	[27:21]	Stop Date--Date when job detached from station (in binary)
Word 8		Attach Time--Time when job attached to station
Word 9		Detach Time--Time when job detached from station

CREATION OF REMOTE LOG ENTRIES. As indicated above, log-in, log-out, and control card entries are made at the time at which they occur. This is possible since the information contained in those entries is immediately available.

The information contained within a Job Statistics entry is accumulated during the time which a remote terminal is attached to a program. The entry is recorded in the Remote Log at the time a program and remote terminal become detached from one another.

The responsibility of dictating which remote station is to be charged for any particular "slice" of a program's processor, I/O, and pro-rated time is strictly that of the object program. The task involved in specifying the station to be charged is, however, an easy one. The procedure involved in slicing times is as follows.

The MCP maintains a table, called USERSTA, which contains one location for each program in the mix. The contents of a given program's location in this table is the station address of the remote station presently specified to be charged for the time used by that program.

When a program enters the mix, its location in the USERSTA table is set to the address of station 0/0, a non-existent remote terminal. (The times assigned to station 0/0 are those which the program does not assign to any given station, i.e., they are unassigned times.) Then from that time until the address in that program's USERSTA location changes, station 0/0 is charged for all processor, I/O, and pro-rated times charged to the program. When the address in the program's USERSTA location changes, the remote terminal whose address is then specified begins being charged for the times assigned to the program, etc.

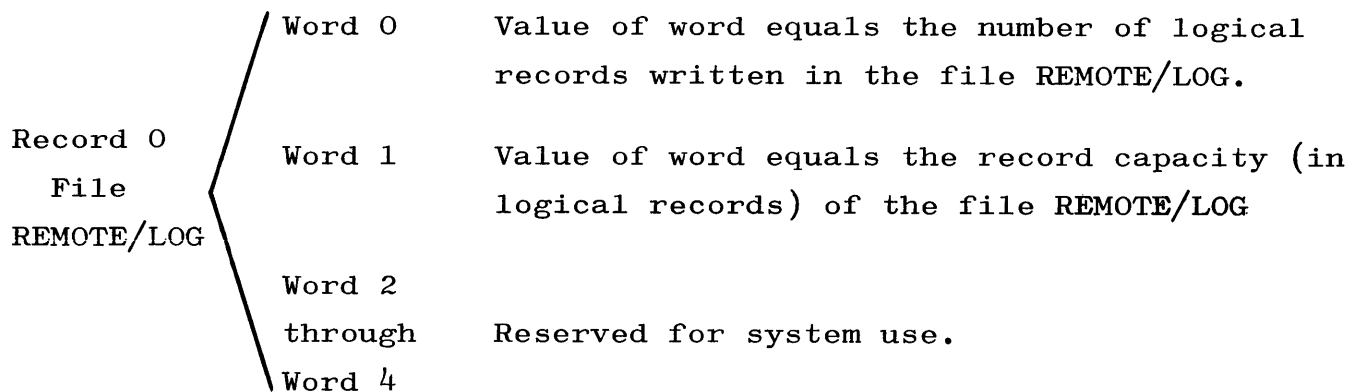
The way in which a program designates the address to be placed in USERSTA -- i.e., the way in which a program designates the station statement referencing the station. (In ALGOL this involves a statement of the form STATUS (TUBUF,0) or STATUS (TUBUF,1); in COBOL it involves a statement such as MOVE FILENAME FROM TU,BUF TO STATUSWORD or MOVE FILENAME FROM TU, BUF AFTER CHECK TO STATUSWORD.) Each time such an interrogate is performed, the MCP checks to see if the terminal buffer address currently in the program's USERSTA location is different from the one specified in the interrogate statement. If it is, the "old station" is charged with all times since the previous change in USERSTA and the new station is established as the new recipient of time.

It should be noted that if a program wishes to designate certain times as being "unassigned" (i.e., assigned to station 0/0), it should perform a passive interrogate on station 0/0.

Whenever a station is "detached" from a program, a job statistic entry is recorded in the log. This entry contains all the times which were allotted to the station in the manner described above.

FILE MAINTENANCE PROCEDURES. To retain information for the file REMOTE/LOG, a FILE Card group should appear in the Cold Start Deck (refer to page 3-23).

The first record of the file REMOTE/LOG (i.e., the record with relative address 0) describes the remainder of the file. Contents of record 0 are:



A user program must initialize word 0 of the file REMOTE/LOG to 0 and word 1 to the record capacity of the file. For example, if the FILE card in the FILE Card group of the Cold Start Deck has the form

FILE REMOTE/LOG,1 (x)1000

then a user program must initialize record 0, word 0 to 0 and record 0, word 1 to 6000.

The operator is notified when the log is half-full and when the log is full. Should the log become full, wraparound occurs.

If the log is not present, the operator is notified the first time the log is accessed.

Operator notification is via the SPO; the messages are:

#REMOTE/LOG FULL

This message is typed when the log is full. Wraparound occurs the next time the log is accessed.

#DUMP REMOTE/LOG

This message is typed when the log is half-full.

#NULL REMOTE/LOG

This message is typed the first time the remote log is accessed and not present.

STATISTICS LOG.

The following paragraphs describe the operation and use of the system measurement facilities as presently implemented on both the standard and time sharing versions of the Master Control Program. The facilities are offered as a compile time option and can be included in a new system by giving the \$ SET Card, with the STATISTICS parameter, the value of TRUE. These facilities, as they presently exist, form portions of a basic software monitor of system performance. Generally, they concentrate on:

- a. Disk activity as it relates to the total system.
- b. Detailed information on resource allocation on a job-to-job basis.

It can be expected that both the configuration and scope of the measurement facilities now provided will change as statistical requisites vary in the future.

GENERAL CHARACTERISTICS.

The overall approach has been to provide a statistical data base of a dual nature, the two portions of which provide a glimpse into the macro- and micro-levels of system utilization. These portions are, at present, complementary; but taken together, they provide a representative picture of system utilization.

One part of this data base is an extended log which contains a detailed look at the actual resources used and system overhead encountered during job execution. Data are collected at various points in the MCP to reflect the various stages of job execution. These are gathered together and entered into the log at job termination. Such a data base provides a picture of the operating environment in which a particular job ran or that environment which existed over a specified period of time.

The other portion of the basic data base exists in the form of a system statistics file which is continuously updated and periodically transferred to permanent disk storage. This master statistics file is updated by information which is either time or task related. Time related information is initiated via the system timer. The nature of the information contained in this file has been, up to now, incremental and cumulative in nature. This information is updated in core and periodically (i.e., whenever NSECOND is called) written on disk for temporary storage. This temporarily stored information is, in turn, periodically (approximately every 30 minutes*) transferred as one record to a permanent system file on disk (SYSTAT <system mnemonic> / DISK) from a temporary system file on disk (SYSTEM <system mnemonic>/STATS); after which the temporary storage area in core is re-initialized to zero and the acquisition of statistics begins anew. For both time sharing and standard versions an empty system statistics file can be created with the SY message entered through the supervisory printer. The name of the current statistics file (i.e., SYSTAT <system mnemonic> / DISK) is changed to <number 1> ON <number 2> / SYSTAT <system mnemonic>; and a new file, SYSTAT <system mnemonic> / DISK, is created, where:

<number 1> is a 2-digit number (00 through 99) of one of the statistics files for a particular date, and

<number 2> is the day created (1 through 365).

* The SI system message allows the adjustment of this value. The format of this message is SI<integer>.

If the MCP involved is compiled with the SHAREDISK \$ SET Option set TRUE, the system designation refers to the system (i.e., SYSTEM A, B, C, or D) for which the file is created. Otherwise, the system designation is blank.

The system statistics file is automatically filled with one record after a 30-minute interval, beginning with the time of the initial Halt/Load of the system. As the SY Message causes subsequent time intervals to begin at the time the message is entered, it can be used to cause the accumulation of statistics for periods beginning on the hour and half hour.

TIME SHARING.

Statistics relating to the time sharing system are kept in a system statistics file and in the TS Log. The log statistics are kept in both the Type 8, EOJ Statistics Message, already present in the TS Log, and a Type 19 Message used for statistics only.

STANDARD SYSTEM.

Statistics relating to the batch system are kept in the system statistics file, previously mentioned with respect to total system usage and in a pseudo log file for individual job statistics. This latter file contains one record per job run on the system, each of which contains some of the information found in the regular system log (processor time, I/O time) in addition to the data obtained through increased job monitoring. This approach is taken to ease off-line analysis of job statistics while not hampering normal log analysis. The pseudo log has been implemented in such a way that an SL Message entered via the supervisory printer causes the current pseudo log file to be saved in much the same way the SY Message causes the system statistics file to be saved. That is, the current statistics log file, STLOG <system mnemonic> / STATS, is changed to <number 1> ON <number 2> / STLOG <system mnemonic>; and a new STLOG <system mnemonic> / STATS is created.

OPERATION.

Once initialized the system begins accumulating statistics without operator intervention. As noted previously the system statistics file

is updated at 30-minute intervals. The statistics log file, if present, contains a new entry for each job initiated on the system. As the statistical information in core is stored in the current SYSTEM <system mnemonic> / STATS File and subsequently re-initialized to zero after an SY Message, this message can be used to gather data for periods of less than 30 minutes. That is, an SY Message entered before and after a specified period of time creates a single record file containing measurement data for that period.

If either statistics file becomes full, the system automatically saves the filled file in the same manner as if either an SY or SL Message has been entered.

FILE DESCRIPTIONS.

SYSTEM STATISTICS FILE. This file is composed of 60-word logical records. The first word of the record following the last statistical record contains a file terminate word in the form of the number @3777777777777777. The contents of both the time sharing and batch files are the same with the exception that information not applicable to the standard system (e.g., data related to swapping) have zero entries in the pertinent elements of the record. The format of a typical record is as follows (entries marked by * pertain only to the time sharing file; word numbers included within parentheses contain the number of disk segments involved for the number of I/O's contained by the word whose number is to the left):

<u>Word</u>	<u>Description</u>
0	Total number of disk I/O operations for time period involved
1	Time since last Halt/Load
2	Total number of disk I/O operations handled by Disk File Controller A
3	Total number of timer interrupts
4	EU 0 disk activity (number of disk I/O operations)

<u>Word</u>	<u>Description</u>
5	EU 1 disk activity (number of disk I/O operations)
6	EU 2 disk activity (number of disk I/O operations)
7	Number of timer interrupts occurring while Disk File Controller (DFC) A is in use
8	Number of timer interrupts occurring while DFC B is in use
9	Number of normal state disk I/O operations
10 (40)	Number of disk I/O operations involving MCP code
11 (41)	Number of disk I/O operations involving ESP code
12 (42)	Number of disk I/O operations involving bypass directory
* 13	Number of disk I/O operations originating below the fence
14	Number of timer interrupts for which the mix is not zero
* 15 (45)	Number of disk I/O operations resulting from swapping
* 16 (46)	Number of disk accesses to DATACOM input/output tanks
17	Number of timer interrupts occurring while both DFC A and DFC B are in use
18 (48)	Number of disk I/O operations resulting from library maintenance
* 19 (49)	Number of disk I/O operations resulting from code below fence
* 20 (50)	Number of disk I/O operations resulting from code above fence

<u>Word</u>	<u>Description</u>
* 21 (51)	Number of disk I/O operations resulting from data below fence
* 22 (52)	Number of disk I/O operations resulting from data above fence
* 23 (53)	Number of disk I/O operations involving SYSTEM/DISK
24 (54)	Number of disk I/O operations involving log
25 (55)	Number of disk I/O operations involving name portion of a directory section
26 (56)	Number of disk I/O operations involving portions of a directory section
27 (57)	Number of disk accesses to program files
28	Date - time sharing system MM/DD/YY - standard system YYDDD
29	Time of day record is entered into file
30	Total number of disk segments involved for all disk I/O operations recorded
31	Number of timer interrupts while mix is not zero
32	Cumulative disk delay, i.e., time from I/O request to I/O initiation
* 33	Cumulative swap delay
* 34	Number of non-zero sway delays
35	Number of disk I/O operations via I/O channel 1
36	Processor idle/busy: number of timer interrupts occurring while MCP is in NOTHINGTODO state

<u>Word</u>	<u>Description</u>
37	Number of disk I/O operations via I/O channel 2
* 38	Number of timer interrupts for which PIMIX is equal to the mix number of CANDE
39	Normal state/disk I/O overlap (number of timer interrupts for which mix is not zero and DFC A and DFC B is in use)
43	Control state/disk I/O overlap
44	Electronics unit 3 disk activity
47	Time of day when data began being gathered for the particular record involved
58	I/O channel 3 disk activity
59	I/O channel 4 disk activity

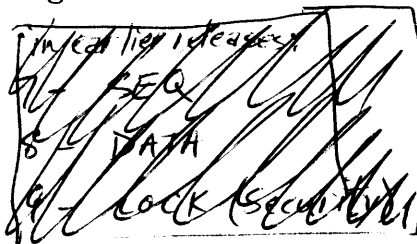
STATISTICS LOG FILE. The statistics log file contains one 15-word logical record for each job begun on the standard system. The first word of the record following the last log entry contains the end-of-file marker as the system statistics file. The format for one record is as follows:

<u>Word</u>	<u>Field</u>	<u>Description</u>
0		Multifile identifier
1		File identifier
2		Starting time
3		Time of job termination
4		Processor time (60ths of a second)
5		I/O time (60ths of a second)
6	[18:15]	Amount of core used by job
7	[33:15]	Amount of core in use by all jobs in mix
8		Job type:

<u>Word</u>	<u>Field</u>	<u>Description</u>
	[42:6]	0 - not a compilation 1 - ALGOL 2 - COBOL 6 - FORTRAN 7 - BASIC 9 - XALGOL 10 - TSPOL
	[36:6]	0 - unknown object program type 1 - BASIC object program 2 - ALGOL object program 3 - COBOL object program 4 - FORTRAN object program 5 - TSPOL object program 6 - XALGOL object program
9		Unused
10	[1:23]	Number of data presence bit interrupts
	[24:24]	Number of code presence bit interrupts
11	[1:23]	Number of data overlays
	[24:24]	Number of code overlays
12		Number of secondary code presence bit interrupts
13		Number of jobs remaining in mix after job termination
14		Unused

TIME-SHARING LOG ADDITIONS. The Type 19 statistics message contains additional job information not found in the Type 8, EOJ statistics, Message. The contents of the Type 19 Message is as follows:

<u>Word</u>	<u>Field</u>	<u>Description</u>
1		Actual time in core
2	[3:15]	DALOC[PIMIX, 0].[33:15]/2 = number of 500-segment sections obtained

<u>Word</u>	<u>Field</u>	<u>Description</u>
	[13:30]	DALOC[PIMIX, (DALOC[PIMIX, 0].[33:15])] = number of 100-segment subsections of last 500-segment section in use by job
3		Time spent in READYQUE
4	[6:6]	(Number of chunks possessed by job) - 1
	[12:6]	Number of last chunk assigned to job
	[13:6]	Number of last chunk assigned to job
	[24:6]	Type of object program: <div style="display: flex; align-items: flex-start;"> <div style="margin-right: 20px;"> <p>1 - BASIC</p> <p>2 - ALGOL</p> <p>3 - COBOL</p> <p>4 - FORTRAN</p> <p>5 - TSPOL</p> <p>6 - XALGOL</p> </div>  </div>
	[30:18]	Creation date of object file
5	[1:23]	Number of forced swaps
	[27:27]	Number of time swaps
6	[1:23]	Number of data presence bit interrupts
	[24:24]	Number of code presence bit interrupts
7	[1:23]	Number of data overlays
	[27:27]	Number of code overlays
8		Number of jobs remaining in mix at job termination
9		Number of secondary code presence bit interrupts

DISK DIRECTORY. See also page 3-20

The MCP maintains, on disk, a Disk Directory which provides information about all permanent files on disk. The Disk Directory is composed of one or more directory sections, depending on the number of files on disk.

Each directory section may contain the directory information required for as many as 15 files. The last segment of a directory section contains the names (i.e., file identifications) of each file defined in the section. The end of the Directory is marked by the first name being equal to 76. Removed files are marked by the first name being equal to 12. The remaining 15 segments are referred to as file headers.

There is one file header for each file defined in the section. Each file header contains various information about the file, such as the creation date, date of last access, etc. Each file header also specifies the number of areas declared for the file, the size of these areas, and the absolute disk address of each area. When a program is using a file, the file header is read into core memory and remains there while the file is being used.

The format of the file header is as follows:

<u>Word</u>	<u>Field</u>	<u>Description</u>
0	[0:15]	Record length
	[15:15]	Block length
	[30:12]	Records per block
	[42:6]	Segments per block
1	[6:18]	Creation date for logging (when on disk)
	[25:23]	Creation time for logging (when on disk)
2	[0:48]	0 - free file
	[1:1]	0 - sole user, public, or private file
	[1:1]	1 - security file
	[6:42]	Primary user's code

<u>Word</u>	<u>Field</u>	<u>Description</u>
3	[1:1]	1 - new file header format
	[2:10]	Save factor (binary)
	[12:18]	Date of last access (binary)
	[30:18]	Creation date (binary)
4	[1:1]	1 - file is being loaded or name is being changed
	[2:1]	1 - file is opened by an exclusive user
	[3:1]	1 - a program is waiting to use the file
	[4:2]	System number of exclusive user
	[6:1]	Used by autoprint to mark a PBD file
	[7:1]	Used to mark pseudo decks that are created on the time sharing system by ZIP WITH FILE-ID
	[9:2]	2 - file is data 3 - file is program 0 - unknown
	[11:1]	File accessed bit
	[12:4]	System file toggles
	[16:5]	Open count 2 for system 0 (A)
	[21:5]	Open count 2 for system 1 (B)
	[26:5]	Open count 2 for system 2 (C)
	[31:5]	Open count 2 for system 3 (D)
[36:6]	0 - type is unknown 1 - type is BASIC 2 - type is ALGOL 3 - type is COBOL	

<u>Word</u>	<u>Field</u>	<u>Description</u>
		4 - type is FORTRAN
		5 - type is TSPOL
		6 - type is XALGOL
		7 - type is SEQ
		8 - type is DATA
		9 - type is LOCK
	[42:6]	Not used
5	[0:42]	0 - free, public, or information file if [43:6] = ?
	[0:48]	0 - sole user or security file
	[1:1]	1 - private file
	[6:42]	multifile ID of security file
6	[0:42]	0 - free or information file if [43:6] = ?
	[0:48]	0 - sole user, public, or security file
	[1:1]	0 - private file
	[6:42]	file ID of security file
7		Number of logical records (EOF pointer)
8		Number of segments per row
9	[1:1]	Toggle 1 for system 0 (A)
	[2:1]	Toggle 1 for system 1 (B)
	[3:1]	Toggle 1 for system 2 (C)
	[4:1]	Toggle 1 for system 3 (D)
	[5:1]	Toggle 2 for system 0 (A)
	[6:1]	Toggle 2 for system 1 (B)
	[7:1]	Toggle 2 for system 2 (C)
	[8:1]	Toggle 2 for system 3 (D)
	[9:5]	Open count 1 for system 0 (A)
	[14:5]	Open count 1 for system 1 (B)

<u>Word</u>	<u>Field</u>	<u>Description</u>
	[19:5]	Open count 1 for system 2 (C)
	[24:5]	Open count 1 for system 3 (D)
	[29:14]	Not used
	[43:5]	Maximum number of rows
10-29		Disk addresses of rows (0 if not assigned)

The Disk Directory begins at segment 1003 and ends at the value of the DIRECT Card in the Cold Start Deck. If it is desired to read the Directory, a permanent file should be placed on disk through use of the Cold Start Deck.

As shown above, the log date is kept in word 1, field [6:18] in binary form. The dates that are kept in the file header are maintained as follows:

- a. When a new disk file is created, the creation date, date of last access, and log date are set to the time the file is opened.
- b. When an existing file is opened, the date of last access is updated.
- c. When a file is loaded from a library tape, the log date is updated.

PRINTER BACKUP INFORMATION.

FORMAT OF A PBT.

A PBT is written in the binary recording mode. The first block on tape is a standard 10-word label. The multifile ID is PBTMCP and the file ID is BACK-UP. The second block on tape is a tape mark. The third and subsequent blocks are 90-word data blocks. The last block written on a PBT is a standard ending label. Preceding this block is a tape mark. If the tape contains more than one print file, there is

no tape mark separating them. Control of the files is maintained through the control word in each data record.

FORMAT OF BLOCKS ON A PB FILE.

Each block of a PB file is 90 words long, containing five logical records. These records are packed in the block in inverted sequence. Therefore, record number 1 is in words 73 through 90, record number 2 in words 55 through 72, and the fifth record in words 1 through 18. The last block of a print file may contain up to four garbage records.

NOTE

If a PBT is to be read or written by a user program, the file declaration must reflect a 90-word record size and a 90-word block size. The program does not function properly if the file declaration indicates a record size of 18 words.

FORMAT OF RECORDS ON A PB FILE.

The first record of a print file is a control record and is identified by the last word of the record being equal to octal 0000004000000000. The format of this record is as follows:

Word 1 - multifile identification of print file

Word 2 - file identification of print file

Words 3 and 4 - name of program which created this print file

Words 5 through 13 - copy of control card which started program
which created this print file

Word 14 - special forms flag

0 - no forms required

1 - forms message

Words 15 through 17 - not used

Word 18 - control word (octal 0000004000000000)

The second record of each file is a copy of the label record for this print file. The control word in this record includes bit [32:1] which indicates a skip to heading (channel 1) after print. The rest of the records in this printer file are data records. The first 17 words of each record is the print line image. The 18th word in each record is a control word which is in the format of a printer I/O descriptor. The format of this descriptor is as follows:

- [0:9] - 554
- [9:9] - number of words to be printed (≤ 17)
- [18:1] - memory inhibit. If ON, blank results.
- [19:1] - on but not used
- [20:1] - end of print file flag. If 1, this is last record of this print file.
- [21:6] - not used
- [27:2] - space information
 - 0 - no space
 - 1 - double space
 - 2 - single space
 - 3 - double space
- [29:4] - skip information
 - = 0 - space paper as per bits [27:2]
 - $\neq 0$ - skip to channel [29:4] after print.
- [33:15] - logical record number of this record

All skipping and spacing takes place after the line is printed. If a paper motion is required prior to printing, a separate print line is written with memory inhibit ON.

FORMAT OF PRINTER BACKUP FILE ON DISK.

Each PBD file (a printer-backup file on disk) has up to 20 900-segment areas. The name of a PBD file is PBD/nnnnrrrr, where nnnn is a serial

number (in BCL) corresponding to the print file (which is incremented when a print file is opened on PBD), and rrr is the serial number of the backup file within the print file (analogous to reel number on a tape file). Thus, each print file may be composed of more than one physical backup file on disk, all with the same nnnn part.

FILE OPENING ACTION.

When a print file is to be opened, the following action occurs:

- a. If the file may go to a printer, the printers are checked for availability and one is used if possible.
- b. If the file may go to a PBT (if an existing PBT is available), it is used; otherwise, if tape is available, a PBT is created and used.
- c. If the file may go to PBD, a PBD is created and used.
- d. If a unit is not found for the file, a message is typed to inform the operator. If a unit of the specified type is made available, it is used. If the operator changes the type with an OU reply, the above process is repeated.

SPECIAL FORMS.

If the print file is opened on a printer-backup file, any special forms requirement is deferred until the backup file is printed. If the print file is opened on a printer:

- a. A printer is chosen.
- b. The operator is informed that special forms are required on that unit by the message # <unit> FM RQD... . The operator may then:
 - 1) Load the forms onto that unit and reply OK.
 - 2) Load the forms onto the other printer, SV the first printer, and reply OU LP.

- 3) Reply OU MT or OU DK to force the chosen printer to be released to open a backup file.

When a backup is printed which requires special forms, the message # FM RQD <unit> FOR <mfid> / <fid> OF <program name> is typed, to which the operator may reply with OK, WY, or DS.

CLOSING A PRINT FILE ON DISK.

When a print file on disk is closed and if the system option autoprint is set, it is scheduled to be printed. If autoprint is not set, a message is typed to inform the operator that a PBD exists and may be printed by the message PBD nnnn REL... .

LOGGING OF PB FILES.

When a print file is printed from a PB file, an entry is made into the log containing the header card information of the program which initially created the print file, and all other appropriate information. The code (in the first word of "General Program Information") is 5, to indicate the printing of a printer backup file.

APPENDIX A
CHARACTER REPRESENTATION

BCL CHARACTER SET	PUNCHED CARD CODE		BINARY CODES		
			A	B	C
			BCL CODE P BA 8421	PAPER TAPE E C L XOH 8421	INTERNAL CODE BA 8421
	ZONE	NUMERIC			
Blank			1 01 0000	00 1 0000	11 0000
.	12	8-3	1 11 1011	11 0 1011	01 1010
[12	8-4	0 11 1100	11 1 1100	01 1011
(12	8-5	1 11 1101	11 0 1101	01 1101
<	12	8-6	1 11 1110	11 0 1110	01 1110
+	12	8-7	0 11 1111	1 00 0 0000	01 1111
&	12		0 11 0000	11 1 0000	01 1100
\$	11	8-3	0 10 1011	10 1 1011	10 1010
*	11	8-4	1 10 1100	10 0 1100	10 1011
)	11	8-5	0 10 1101	10 1 1101	10 1101
;	11	8-6	0 10 1110	10 1 1110	10 1110
<	11	8-7	1 10 1111	10 0 1111	10 1111
-	11		1 10 0000	10 0 0000	10 1100
/	0	1	0 01 0001	01 1 0001	11 0001
,	0	8-3	0 01 1011	01 1 1011	11 1010
%	0	8-4	1 01 1100	01 0 1100	11 1011
=	0	8-5	0 01 1101	01 1 1101	11 1101
]	0	8-6	0 01 1110	01 1 1110	11 1110
"	0	8-7	1 01 1111	01 0 1111	11 1111
#		8-3	1 00 1011	00 0 1011	00 1010
@		8-4	0 00 1100	00 1 1100	00 1011
:		8-5	1 00 1101	00 1 1101	00 1101
>		8-6	1 00 1110	00 0 1110	00 1110
~		8-7	0 00 1111	00 1 1111	00 1111

APPENDIX A (cont)
CHARACTER REPRESENTATION

BCL CHARACTER SET	PUNCHED CARD CODE		BINARY CODES		
			A	B	C
	ZONE	NUMERIC	BCL CODE P BA 8421	PAPER TAPE E C L XOH 8421	INTERNAL CODE BA 8421
+	12	0	0 11 1010	11 1 1010	01 0000
A	12	1	1 11 0001	11 0 0001	01 0001
B	12	2	1 11 0010	11 0 0010	01 0010
C	12	3	0 11 0011	11 1 0011	01 0011
D	12	4	1 11 0100	11 0 0100	01 0100
E	12	5	0 11 0101	11 1 0101	01 0101
F	12	6	0 11 0110	11 1 0110	01 0110
G	12	7	0 11 0111	11 0 0111	01 0111
H	12	8	1 11 1000	11 0 1000	01 1000
I	12	9	0 11 1001	11 1 1001	01 1001
x	11	0	1 10 1010	10 0 1010	10 0000
J	11	1	0 10 0001	10 1 0001	10 0001
K	11	2	0 10 0010	10 1 0010	10 0010
L	11	3	1 10 0011	10 0 0011	10 0011
M	11	4	0 10 0100	10 1 0100	10 0100
N	11	5	1 10 0101	10 0 0101	10 0101
O	11	6	1 10 0110	10 0 0110	10 0110
P	11	7	0 10 0111	10 1 0111	10 0111
Q	11	8	0 10 1000	10 1 1000	10 1000
R	11	9	1 10 1001	10 0 1001	10 1001
≠	0	8-2	1 01 1010	01 0 1010	11 1100
S	0	2	0 01 0010	01 0 0010	11 0010
T	0	3	1 01 0011	01 0 0011	11 0011
U	0	4	0 01 0100	01 1 0100	11 0100
V	0	5	1 01 0101	01 0 0101	11 0101
W	0	6	1 01 0110	01 0 0110	11 0110
X	0	7	0 01 0111	01 1 0111	11 0111
Y	0	8	0 01 1000	01 1 1000	11 1000
Z	0	9	1 01 1001	01 1 1001	11 1001

APPENDIX A (cont)
CHARACTER REPRESENTATION

BCL CHARACTER SET	PUNCHED CARD CODE		BINARY CODES		
			A	B	C
			BCL CODE P BA 8421	PAPER TAPE E C L XOH 8421	INTERNAL CODE BA 8421
	ZONE	NUMERIC			
0		0	0 00 1010	01 0 0000	00 0000
1		1	1 00 0001	00 0 0001	00 0001
2		2	1 00 0010	00 0 0010	00 0010
3		3	0 00 0011	00 1 0011	00 0011
4		4	1 00 0100	00 0 0100	00 0100
5		5	0 00 0101	00 1 0101	00 0101
6		6	0 00 0110	00 1 0110	00 0110
7		7	1 00 0111	00 0 0111	00 0111
8		8	1 00 1000	00 0 1000	00 1000
9		9	0 00 1001	00 1 1001	00 1001
?	All Other Card Codes		0 00 0000	00 0 1101	00 1100

NOTES

1. Characters are listed in collating sequence. Blank is lowest and the invalid code (?) is highest.
2. Internal code 00 1100 (?) is punched as card code 8-2.
3. Paper tape feed punches are 11 1 1111.
4. A paper tape feed code acts as a delete code and when read is not transferred to the associated processor.
5. The paper tape sprocket hole is between channels 8 and 4.

APPENDIX B
IDENTIFIERS

Identifiers which may be required by control cards or program-parameter cards for modification to any of the programs contained in the system are as follows:

ALGOL Compiler

Program Identifier	ALGOL
Program Identifier Suffix	DISK
File ID's	
Source program card input file ID	CARD*
Source program tape input file ID	TAPE
Source program tape output file ID	NEWTAPE
Source program printer output file ID	LINE
Source program punch output file ID	PNCH

COBOL Compiler

Program Identifier	COBOL
Program Identifier Suffix	DISK
File ID's	
Source program card input file ID	CARD
Source program tape input file ID	TAPE
Source program tape output file ID	NEWTAPE
Source program printer output file ID	LINE

FORTRAN Compiler

Program Identifier	FORTRAN
Program Identifier Suffix	DISK
File ID's	
Source program card input file ID	CARD
Source program tape input file ID	TAPE
Source program tape output file ID	NEWTAPE
Source program printer output file ID	LINE

* It should be noted that the ALGOL, COBOL, and FORTRAN Compilers will use the card file immediately following the COMPILE card, regardless of file ID.

APPENDIX B (cont)
IDENTIFIERS

FORTRAN Translator

Program Identifier	FORTRAN
Program Identifier Suffix	DISK

ESPOL Compiler

Program Identifier	ESPOL
Program Identifier Suffix	DISK

File ID's

Source program card input file ID	CARD
Source program tape input file ID	TAPE
Source program tape output file ID	NEWTAPE
Source program printer output file ID	LINE
Card output file for compiled code	DECK
Disk output file for compiled code	DISK

File ID's

FORTRAN program card input file ID	CARD
Generated ALGOL program tape output file ID	PUNCH
Generated ALGOL program punch output file ID	PPUNCH
Generated ALGOL program print output file ID	PRINT

APPENDIX C
MESSAGES

GENERAL.

The operator and the MCP communicate with each other by means of the supervisory printer and keyboard. Through the use of the supervisory printer, the MCP can direct the operator and supply the answers to inquiries from him. The operator, on the other hand, can acknowledge instructions typed by the MCP and initiate inquiries that must be answered by the MCP.

SYSTEM MESSAGES.

The messages given to the operator are of two basic types: those for informative purposes only and those requiring action by the operator. To minimize the amount of time used by the supervisory printer, the messages are made up of mnemonic codes followed by the variable information that is needed to make the message meaningful. Each element of the message (including the mnemonic code) is separated from adjacent elements by at least one blank.

A system message which requires an action by the system operator is prefixed with the character #.

System messages which denote that a program is to be discontinued before EOJ are preceded by the character -.

System messages related to the breakout and restart facility are preceded by the character pair --.

In the descriptions of system messages, the construct <job specifier> is used and is defined as follows:

<program specifier> = <mix index>

An example of a <job specifier> is:

PROGID/SUPID=1

The <mix index> provided in a <job specifier> is one to be used in

APPENDIX C (cont)

MESSAGES

any keyboard input messages referencing the subject program if the input message requires a $\langle \text{mix index} \rangle$.

Another construct which is used in describing keyboard output messages is $\langle \text{termination reference} \rangle$ which is defined as follows:

$$S = \langle \text{integer} \rangle, A = \langle \text{integer} \rangle$$

where the $\langle \text{integer} \rangle$ following the S is the number of the program segment being executed when the subject program is discontinued (except in the case of an intrinsic segment where the number refers to the last nonintrinsic segment executed). The $\langle \text{integer} \rangle$ following the A is the relative address, within the segment specified, of the syllable last executed.

A third construct is $\langle \text{file specifier} \rangle$ which is defined as follows:

$$\langle \text{file identification prefix} \rangle / \langle \text{file identification} \rangle$$

or

$$\langle \text{program identifier} \rangle / \langle \text{program identifier suffix} \rangle$$

The construct $\langle \text{time} \rangle$ is defined as the time of day as reckoned on a 24-hour clock.

The construct $\langle \text{remote specifier} \rangle$ has the following form:

$$\begin{aligned} &\langle \text{terminal unit number} \rangle / \langle \text{buffer number} \rangle \\ &\langle \text{terminal unit number} \rangle ::= \text{unsigned integer} \\ &\langle \text{buffer number} \rangle ::= \text{unsigned integer} \end{aligned}$$

Both the $\langle \text{terminal unit number} \rangle$ and the $\langle \text{buffer number} \rangle$ have a field width of two digits.

The construct $\langle \text{error condition} \rangle$ is defined as follows:

$$\begin{aligned} &\langle \text{disk parity} \rangle \mid \langle \text{tape parity} \rangle \langle \text{invalid record size on tape} \rangle \\ &\langle \text{missing label on tape} \rangle \end{aligned}$$

APPENDIX C (cont)

MESSAGES

The construct <unit mnemonic> is defined as any of the three-character codes listed below. The definition of each unit mnemonic, as recognized by the MCP, is listed to the right of the code.

<u>Code</u>	<u>Definition</u>
MTA	Magnetic tape unit A
MTB	Magnetic tape unit B
MTC	Magnetic tape unit C
MTD	Magnetic tape unit D
MTE	Magnetic tape unit E
MTF	Magnetic tape unit F
MTH	Magnetic tape unit H
MTJ	Magnetic tape unit J
MTK	Magnetic tape unit K
MTL	Magnetic tape unit L
MTM	Magnetic tape unit M
MTN	Magnetic tape unit N
MTP	Magnetic tape unit P
MTR	Magnetic tape unit R
MTS	Magnetic tape unit S
MTT	Magnetic tape unit T
MTX	All scratch tapes
CRA	Card reader A
CRB	Card reader B
LPA	Line printer A
LPB	Line printer B
CPA	Card punch A
PPA	Paper tape punch unit A
PPB	Paper tape punch unit B
PRA	Paper tape reader A
PRB	Paper tape reader B
SPO	Supervisory printer
CDA through CDZ, excluding CDI and CDO, and CD2 through CD9	Pseudo card readers A through Z, excluding I and O, and 2 through 9

APPENDIX C (cont)

MESSAGES

<u>Code</u>	<u>Definition</u>
DCA	Data communications control unit
DKA	Disk control A
DKB	Disk control B

The construct <system mnemonic> refers to designations given to entire systems, especially under Sharedisk. Given below is the definition of each system mnemonic as utilized by the MCP. System mnemonic codes for input messages, output messages, and the internal method of system designation used by the MCP are listed to the left of each definition.

<u>Input Code</u>	<u>Output Code</u>	<u>Internal* Code</u>	<u>Definition</u>
SYA	A	0	First system under Sharedisk
	blank		Any system not under Sharedisk
SYB	B	1	Second system under Sharedisk
SYC	C	2	Third system under Sharedisk
SYD	D	3	Fourth system under Sharedisk

KEYBOARD OUTPUT MESSAGES.

The keyboard output messages, or system messages, are listed and described in the following paragraphs.

<job specifier> = <mix index> ACCEPT

This message is typed on the SPO for a programmatic operator input request.

*ABORT LOG DESTROYED
MCP is reporting that it cannot find
an abort log of programs that had been
aborted previously*

-ACTV INTRGT TU 0 <prog. id.> / <prog. id.> = <mix index>

This message notifies the operator that a program attempting an active interrogate on terminal unit 0 has terminated.

* Internally, Systems A, B, C, and D, respectively, are always referred to as 0, 1, 2, and 3 except in the case of disk segment zero where they are respectively referenced as Systems 1, 2, 3, and 4.

APPENDIX C (cont)

MESSAGES

<priority>:<job specifier> = <mix>:BADUMP ON <unit>

The system cannot copy a break file onto the indicated tape. It will try again on a new reel.

BED OVRFLW

The occurrence of this message denotes that too many entries have been made in the BED, a table used by the control section of the MCP. If the condition indicated by this message should occur, a HALT-LOAD operation is required.

<job specifier> = <mix index> BOJ <time> FROM <remote specifier>

This message is typed when an object program first begins to execute, providing the TYPE BOJ option is set. The FROM <remote specifier> is included if the job is executed from a remote station with SPO capabilities.

<compiler name> / <program identifier> = <mix index> BOJ <time>
FROM <remote specifier>

This message is typed when either the ALGOL or the COBOL Compiler begins a compilation, providing the TYPE BOJ option is set. The FROM <remote specifier> is included if the compile is executed from a remote station with SPO capabilities.

<priority>:<job specifier> = <mix>:BREAKnn BUILT

The specified job just broke, creating the break file <program name> / BREAKnn. The break file is then moved to an output tape, if necessary.

<unit mnemonic> BUSY

The occurrence of this message denotes that an I/O operation was attempted on the specified unit, and the unit was found to be apparently busy.

--CAN-T BREAK <data file designator> <rdc>:<job specifier>

The job tried breaking with a file of unsuitable type open. The break try is ignored.

APPENDIX C (cont)

MESSAGES

CONTROL CARD ERROR <unit mnemonic> {information from control card}
The occurrence of this message indicates that the MCP has expected to read control information from the designated I/O unit but has found the information to be in error.

CP RQD <data file designator> <rdc> : <job specifier>
The occurrence of this message indicates that a program has need for a card punch and no such I/O device is currently available.

DATACOM / INQUIRY INTERRUPT IGNORED BY MCP
This message is typed on the SPO if a data communications interrupt is received and the MCP has been compiled with the DATACOM and INQUIRY options set FALSE.

<unit mnemonic> / <I/O operation> DA = <integer>; #SEG = <integer>;
#RTRY = <integer>; #TRNS = <integer>

This message is typed when retries had to be made on the disk file.

■ The <I/O operation> is an R if it was on a read, W if on a write. The number appearing after DA is the disk address. The number appearing after #SEG is the number of segments read or written. The number appearing after #RTRY is the number of retries necessary (modulo 10). If this number is equal to 0, a successful retry was not made. The number appearing after #TRNS is the number of disk transactions since the last HALT-LOAD operation.

-DC TU NOT OUTPUT POSSIBLE <job specifier>. <termination reference>
The occurrence of this message denotes that an object program attempted to perform a write on a terminal unit that was not set for output. Because of this erroneous action, processing of the object program was discontinued.

-DEC ERR : ARRAY DIMENSION = <integer>
<job specifier>, <terminal reference> is typed if an array with an illegal row size is declared.

APPENDIX C (cont)

MESSAGES

-DEC ERR : NO. DISK ROWS = \langle integer \rangle ,
 \langle job specifier \rangle , \langle terminal reference \rangle is typed if a disk file is
declared with more than 20 rows.

DECK # \langle integer \rangle IN USE BY SYSTEM \langle system mnemonic \rangle

This message is typed when the operator of one system tries to re-
move a control deck in use by another system.

DECK \langle integer \rangle REMOVED

This message is typed when a control deck is removed from the disk
because of the completion of the job or a keyboard input message.

-DIMENSION SIZE ERR \langle job specifier \rangle , \langle terminal reference \rangle

A BASIC program attempted to perform a matrix operation in which the
resulting matrix is too small to contain the result of the matrix
operation. The program is discontinued.

ODIRERR

The occurrence of this message denotes that the MCP has detected a
bad entry in the Disk Directory. The job accessing the file which
has the bad entry is put to sleep permanently by the MCP, while the
remaining jobs in the MIX are allowed to attempt to go to EOJ. The
system slowly comes to a halt. Jobs should not be initiated after
the occurrence of the above message. A COOL START should be per-
formed, and the system should then be HALT/LOADed.

. \langle mfid \rangle / \langle fid \rangle DISK ADDRESS ERROR

This message occurs when conflicting disk addresses are found in
file headers.

The file which caused the above message to be elicited may have
valid addresses in the file header. A file previously processed by
the complementing algorithm may have encroached upon disk space
occupied by other file(s), thus causing the error. However, receipt
of the above message indicates that directory destruction has occur-
red and the user should salvage as many files as possible.

APPENDIX C (cont)

MESSAGES

DISK FAILURE - <unit mnemonic>

If this message occurs and is not followed by a ...#RTRY=... message, a HALT-LOAD must be performed.

DKA(R) DA=nnnnnnnn; #SEG=mm; #RTRY=x; #TRNS=yyyyyyyyyy; MIX=zz

DKA(W) DA=nnnnnnnn; #SEG=mm; #RTRY=x; #TRNS=yyyyyyyyyy; MIX=zz

DKB(R) DA=nnnnnnnn; #SEG=mm; #RTRY=x; #TRNS=yyyyyyyyyy; MIX=zz

DKB(W) DA=nnnnnnnn; #SEG=mm; #RTRY=x; #TRNS=yyyyyyyyyy; MIX=zz

n = absolute disk address in decimals.

m = number of segments required before successful completion.

x = number of retries. 1-9 indicates a successful retry.

0 indicates that a successful retry was not accomplished.

y = number of read/writes in n area since the last HALT/LOAD.

z = mix number of program which performed the I/O operation.

This message indicates an area on disk was accessed which caused the I/O error routines to attempt retries. This message generally occurs in conjunction with the DISK FAILURE message. Information in this message should be retained for analysis by the field engineers.

-DIV BY ZERO <job specifier> , <termination reference>

The occurrence of this message denotes that an object program performed a Divide operation using a zero denominator. Consequently, processing of the subject program was discontinued.

DIV BY ZERO BRANCH <job specifier> , <termination reference>

This message is typed upon the occurrence of a Divide by Zero when the programmatic recovery feature is being used.

<job specifier> = <mix index> DS-ED

This message is typed if processing of an object program is discontinued before End-of-Job, providing the EOJ option is set.

<compiler name> / <program identifier> = <mix index> DS-ED

This message is typed if a compilation is discontinued before the

APPENDIX C (cont)

MESSAGES

compiler has reached End-of-Job, providing the TYPE EOJ option is set.

DT PLEASE

This message is typed at HALT-LOAD time if the TYPE DATE option has been set. The system operator is required to enter a DT message before processing can commence.

DUMP REMOTE/LOG

This message is typed when the log is half-full.

<file specifier> DUMPED <unit mnemonic>

This message is typed after the MCP has performed an operation specified on a DUMP control card.

DUP FIL <data file designator> <rdc> : <job specifier> <duplicate file list>

The occurrence of this message denotes that an object program wishes to open an input file and that the MCP has found more than one file with the desired identification. Files on disk are not taken into regard. The duplicate-file condition causes the designated program to be suspended until operator action is taken. The condition may be rectified by making only one of the acceptable files available and then entering a <mix index> OK message.

ADDITIONAL USE OF THE IL MESSAGE

The IL message may now be used to designate the file to be opened when the DUP FILE condition arises.

DUP LIBRARY <file specifier> : <job specifier>

The occurrence of this message indicates that an attempt has been made to add a file to the disk library, but the file's name is identical to the name of a file already in the Disk Directory. The program which attempted to add the file to the library is temporarily suspended until the operator remedies the situation. To remedy the situation, the system operator may eliminate the conflict by using a CHANGE card or REMOVE card and then an OK message, or he may DS

APPENDIX C (cont)

MESSAGES

the program that attempted to place the new file in the library, or he may enter an RM keyboard input message.

-EOF NO LABEL <file designator> : <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has reached the end of the designated input file and has not specified what is to be done. Consequently, processing of the program is discontinued.

<job specifier> = <mix index> EOJ

This message is typed when an object program reaches End-of-Job, providing the TYPE EOJ option is set.

<compiler name> / <program identifier> = <mix index> EOJ

This message is typed when a compiler reaches End-of-Job, providing there were no syntax errors and providing the TYPE EOJ option is set.

-EOT NO LABEL <file designator> : <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has reached the end of the designated file's declared area, as on disk. Consequently, processing of the program is discontinued.

ESPDISK ERROR

The occurrence of this message indicates that an error has occurred in the handling of the MCP's scratch disk, ESPDISK. If the condition indicated by this message should occur, a HALT/LOAD operation is required.

-EXCESS TIME <job specifier> , <termination reference>

The occurrence of this message denotes that the process time of an object program has exceeded the time specified on its PROCESS program-parameter card. Consequently, processing of the program is discontinued.

APPENDIX C (cont)

MESSAGES

<file specifier> EXPIRED

This message is typed in reference to files on disk in response to the EX input message if (the file's date of last access) + (the file's SAVE factor) does not result in a date greater than the current date.

-EXPON OVRFLW <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has performed an operation which caused an exponent overflow to occur. Consequently, processing of the program is discontinued.

EXPON OVRFLW BRANCH <job specifier> , <termination reference>

This message is typed upon the occurrence of an exponential overflow when the programmatic recovery feature is being used.

FACTOR = x, MAX CORE = y, USING z

The MCP responds to a TF message with this message. The letter x is the Factor; y is the actual number of core cells available to object programs, multiplied by the Multiprocessing Factor; and z is the sum of the core requirements of the jobs actually running.

-FAE , <file specifier> . <file attribute identifier> := {value} ,
T = <output media digit> <job specifier> <terminal reference>

This message indicates a file attribute error. The {value} assigned to a file attribute in a file attribute assignment statement has violated the prescribed bounds. (Refer to the Extended ALGOL Reference Manual for a list of the attributes and their corresponding value bounds.) Numeric {value}s within the range of $[-10^7, 10^8]$ are displayed as a signed integer; others are displayed as a signed field of asterisks. Alphanumeric values, e.g., MFID, are displayed as an 8-character alpha.

-FAE , <file specifier> . <file attribute identifier> , {message}
<job specifier> <terminal reference>

This message indicates a file attribute error. During the execution of a file attribute assignment statement, an exceptional condition

APPENDIX C (cont)

MESSAGES

has occurred relating to file disposition, file usage, and so forth. The {message}s are:

a. MYUSE = CANTUSE

The user has attempted to set an attribute, other than MYUSE, and MYUSE is CANTUSE.

b. NOT RWND/CLSD

The file attribute is fixed, and the file has not been (at least) rewound or CLOSED.

c. NOT CLOSRELES

The file attribute is fixed, and the file has not been (at least) CLOSE-RELEASED or LOCKed.

d. CLS*, NOT ALTR

The file attribute is fixed, the close performed prior to the assignment statement was a CLOSE (*), the file is tape, and the user is attempting to alter the TYPE or MFID.

-FAE , <file specifier> . MYUSE , {message} <job specifier>
<terminal reference>

This message indicates a file attribute error. A READ or WRITE statement is performed on a file where MYUSE restricts the I/O.

The {message}s are:

a. TRIED READING

MYUSE is CANTUSE or OUTPUT, and an input from the file has been attempted.

b. TRIED WRITING

MYUSE is CANTUSE or INPUT, and an output to the file has been attempted.

<unit mnemonic> <read-write flag> FAILURE - D <integer>

This message indicates that one of the following error conditions persisted after ten retries:

APPENDIX C (cont)

MESSAGES

<integer> = 19 - parity error between I/O control and core or disk file control.

<integer> = 20 - parity error on transfer from disk.

-FILE UNOPENED <job specifier> , <terminal reference>

The occurrence of this message denotes that an object program attempted to write on a file that has not been opened. Consequently, processing of the program has been discontinued.

-FLAG BIT <job specifier> , <terminal reference>

The occurrence of this message denotes that an object program has performed an operation which caused a word with a flag bit of 1 to be accessed as if it were an operand. Consequently, processing of the program has been discontinued.

FLAG BIT BRANCH <job specifier> , <terminal reference>

This message is typed upon the occurrence of a flag bit when the programmatic recovery feature is being used.

#FM RQD <data file designator> <rdc> : <job specifier>

The occurrence of this message indicates that a program is ready to open a file which, as specified on a label equation card, is required to use special forms. The FM message must be entered before the subject program can continue processing.

-FMT ERR NO LBL <job specifier> , <terminal reference>

The occurrence of this message denotes that the object program tried to evaluate the editing phrase of an ALGOL format using a negative or undefined list element to evaluate a dynamic format phrase. An action label has not been supplied. Consequently, processing of the program has been discontinued.

<job specifier> GONE <time>

The job is a restart which is ESed or DSed before having restarted.

APPENDIX C (cont)

MESSAGES

-H/L MARK DCMCP <Roman numeral> . <integer> MODS RRRRRRRR-

This message is typed immediately following a HALT/LOAD operation. The Roman numeral identifies the level of the MCP, and the integer indicates the number of changes to the basic level. An @ appearing in the string of R's indicates a memory module that is not ready.

<unit mnemonic> IN <data file designator> <rdc> : <job specifier>

This message is typed when a program opens a card or tape file for input, providing the necessary options have been set. The message is typed for object program files if the TYPE OPN option is set. The message is typed for compiler files if both the TYPE OPN and TYPE CMPLRFIL options are set.

-INTGR OVRFLW <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has performed an operation which has caused an integer overflow to occur. Consequently, processing of the program is discontinued.

INTGR OVRFLW BRANCH <job specifier> , <termination reference>

This message is typed upon the occurrence of an integer overflow when the programmatic recovery feature is being used.

-INVALID ADRSS <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has performed an operation which has addressed a memory location in an absent memory module or an address less than 00512. Consequently, processing of the program is discontinued.

INVALID ADRSS

The occurrence of this message denotes that an invalid address has occurred during processing in control state, and the invalid address cannot be associated with a particular program in the mix. If the condition indicated by this message should occur, a HALT-LOAD operation is required.

INVALID LINK

The occurrence of this message denotes that an invalid memory link

APPENDIX C (cont)

MESSAGES

has been detected by the MCP. If the condition indicated by this message occurs, an H/L operation is required.

INVALID ARG CONCAT <job specifier> , <terminal reference>

This message announces that execution of the FORTRAN intrinsic function CONCAT is terminated if the necessary conditions are not satisfied. The function provides general partial word facilities and is referenced as follows:

CONCAT (A, B, S1, S2, N)

where A and B are any integer or real expressions; S1, S2, and N are integer expressions. S1, S2, and N must satisfy the following conditions:

$$S1 > 0$$

$$S2 > 2$$

$$N > 0$$

$$S1 + N < 48$$

$$S2 + N \leq 48$$

If these conditions are not satisfied, execution is terminated.

The value of the function is the concatenation of A and B as specified by S1, S2, and N. N bits are taken from B, starting at bit S2, to replace N bits of A, starting at bit S1. This has the same effect as the ALGOL expression A and B [S1:S2 = N], assuming S1, S2, and N to be arbitrary expressions in ALGOL. The following ALGOL constructs can be performed with CONCAT:

<u>ALGOL</u>	<u>FORTRAN</u>
C.[12:6]←"X";	C = CONCAT (C, "X", 12, 42, 6)
C←A&B[18:45:3];	C = CONCAT (A, B, 18, 45, 3)
C←B.[36:6]	C = CONCAT (C, B, 42, 36, 6)

Note that, as in ALGOL, bit 0 (the flag bit) cannot be included in either field.

APPENDIX C (cont)

MESSAGES

-INVALID EOJ <job specifier> , <termination reference>

The occurrence of this message denotes that a COBOL program has attempted to execute the END-OF-JOB statement. Consequently, processing of the program is discontinued.

-INVALID INDEX <job specifier> , <terminal reference> , EFF INX IS -
<index value>

The occurrence of this message denotes that an object program has attempted to index out of the limits (in a negative direction) of the array being referenced. Processing of the program is discontinued. The construct <index value> has a maximum size of eight digits.

-INVALID INDEX <job specifier> , <terminal reference> , <index value> GEQ <descriptor size field>

The occurrence of this message denotes that an object program has attempted to index out of the limits (in a positive direction) of the array being referenced. Processing of the program is discontinued. The constructs <index value> and <description size field> have a maximum size of four digits.

INVALID INDEX BRANCH <job specifier> , <termination reference>

This message is typed upon the occurrence of an invalid index when the programmatic recovery feature is being used.

-INVALID INPUT DATUM <job specifier> , <terminal reference>

The data typed in response to an INPUT statement in a BASIC program is not of proper form to satisfy the input list. The program is discontinued.

-INVALID PRL <job specifier> , <terminal reference>

The occurrence of this message indicates that either:

- a. The DSKTOG (OPTN 28) is set and an object program has tried to access an absolute disk address below the user disk area; or

APPENDIX C (cont)

MESSAGES

- b. The RELTOG (OPTN 27) is set and an ALGOL object program has attempted to perform a RELEASE statement referencing disk.

In either case, processing of the program is discontinued.

<unit mnemonic> INV CHR IN COL <integer>

This message is typed when a card has an invalid character in a column other than column 1 of a control card. The column with the invalid character is given in the message. The operator must replace this card with a correct card.

INV KBD {typed-in information}

This message is typed if the MCP does not recognize a message entered from the keyboard.

-INV STN <tu> <buff>

This message is typed if the MCP does not recognize the remote station address used in a remote terminal message.

I/O ERROR <integer> <file designator> : <job specifier>

There are a number of messages which have the above format; the <integer> in the message denotes its specific meaning. The meanings of this message are listed below according to the <integer>.

<u><integer> Value</u>	<u>Meaning</u>
1	A COBOL program has attempted to open an input file that is not closed; consequently, processing of the program is discontinued.
3	A COBOL program has attempted to open-reverse a file that is not closed; consequently, processing of the program is discontinued.
5	A COBOL program has attempted to open-reverse a file that is not blocked properly; consequently, processing of the program is discontinued.

APPENDIX C (cont)

MESSAGES

<u><integer></u> <u>Value</u>	<u>Meaning</u>
6	A COBOL program has attempted to open an output file that is not closed; consequently, processing of the program is discontinued.
11	An attempt has been made to close an input file which is closed or has never been opened.
12	An attempt has been made to close an output file which is closed or has never been opened.
15	An attempt has been made to read a file for which AT END has already been processed.
16	The record count on an input tape does not agree with the internally accumulated record count. The external record or block count is printed out first in the error message; then the internal record or block count is printed.
17	The block count on an input tape does not agree with the internally accumulated block count. The external record or block count is printed out first in the error message; then the internal record or block count is printed.
18	The HASH TOTAL on a COBOL input tape does not agree with the internally accumulated HASH TOTAL.
19	An irrecoverable parity error has occurred during reading of a file assigned to disk or tape. The message is typed once for each block which is in error unless a USE procedure has been specified. The USE procedure (if any) is executed and control is transferred to the statement following the READ statement.

APPENDIX C (cont)

MESSAGES

<u><integer> Value</u>	<u>Meaning</u>
20	An irrecoverable parity error has occurred on an output tape or disk file. The USE procedure has been executed, allowing programmatic closing of files which must be saved, and the program is now being DSed.
21	An attempt has been made to READ from a file opened as OUTPUT. The program is DSed.
22	An attempt has been made to read from a row of a disk file which has never been created. To get this error, the record number must be less than the highest record number written and greater than 1. When a RANDOM file is written, but records fall only in rows 1 and 3 of a 3-row file, attempts to access records in row 2 cause I/O ERROR 22 instead of executing INVALID KEY statements. A row of disk space is assigned, and the appropriate record is made available to the using program. The contents of the record made available is, of course, unpredictable.
23	An attempt is made to WRITE on a file which has been opened as INPUT. The program is DSed.
24	An attempt is made to WRITE on a file which has been opened REVERSED. The program is DSed.
25	Improper code is passed to the COBOLIO intrinsics. The program is DSed.
26	A block of less than eight characters has been read, or a zero record size has been encountered

APPENDIX C (cont)
MESSAGES

<u><integer></u> <u>Value</u>	<u>Meaning</u>
26 (cont)	during the reading of a TECHNIQUE-B or TECHNIQUE-C file which utilizes the SIZE DEPENDING option. The program is DSed.
27	Not used.
28	While operating under Time Sharing, a SEEK has been issued for a data communications device. The program is DSed.
29	An irrecoverable parity error has occurred while reading a tape file which has been opened REVERSED. The message is typed once for each block which is in error unless a USE procedure has been specified. The USE procedure (if any) is executed, and control is transferred to the statement following the READ statement.
30	Not used.
31	An attempt is made to READ from a file which is closed or has never been opened. The program is DSed.
32	An attempt is made to WRITE to a file which is closed or has never been opened. The program is DSed.
33	An attempt is made to SEEK on a file which is closed or has never been opened. The program is DSed.
34	An attempt is made to WRITE BLOCK on an INPUT file. The program is DSed.

APPENDIX C (cont)

MESSAGES

<u><integer> Value</u>	<u>Meaning</u>
35	An attempt is made to WRITE BLOCK on a file opened REVERSED. The program is DSed.
36	Not used.
37	An attempt is made to WRITE BLOCK on a file which is closed or has never been opened. The program is DSed.
69	An attempt is made to write on disk at an address less than 100. The program hangs in a DO UNTIL FALSE loop. The program may be DSed by the operator.
71	The number of records within a string on a tape, used by a COBOL SORT program, is wrong. This is due to an incorrect Read or Write on that tape. Consequently, processing of the program is discontinued.
72	Parity or blank tape in the sort.
74	Parity or blank tape in the merge.
76	An error has occurred within a string being written by a COBOL SORT program; the number of records that should have been written do not equal the number written on the designated unit. Consequently, processing of the program is discontinued.
79	The number of records that should have been read from other tape units in the final merge pass of a SORT, being performed by a COBOL SORT program, does not equal the number of records written onto the final output tape. However, after action

APPENDIX C (cont)

MESSAGES

<u><integer></u> <u>Value</u>	<u>Meaning</u>
79 (cont)	has been taken to type this message, the SORT has closed the final output reel or has executed the user's output routine, signaling End-of-File. Consequently, the output tape may be used in spite of this error message. The tape unit indicated in this message is meaningless.
80	The total number of records entered as input to the SORT, being performed by a COBOL SORT program, is not equal to the number of records produced as output from the SORT in the final merge pass. However, after action has been taken to write this message, the SORT has closed the final output file or has executed the user's output routine signaling End-of-File. Consequently, the output tape may be used in spite of this message. The tape unit indicated in this message is meaningless.
81	The amount of disk available is insufficient for a disk-only or ITD mode sort.
82	The number of records read from the input does not match the number written to the final output.
83	A disk file has been passed as an output file which is not large enough to hold all of the sorted output data.
84	Disk-only mode. The amount of disk specified is insufficient to do a disk-only sort.
85	ITD mode. The number of records read from a string on tape is not the same number written.

APPENDIX C (cont)

MESSAGES

<u><integer> Value</u>	<u>Meaning</u>
86	Records have not been passed to either a COBOL or an ALGOL SORT program.
87	A sort record description is greater in length than the record description of a file which is passed as an output file to a COBOL sort.

<unit mnemonic> I/O INV ADDR

The occurrence of this message denotes that an invalid address has occurred when data is to be transferred between an I/O channel and core memory. The MCP recognizes this error condition and rectifies the errors if possible. The primary purpose of this message is to draw attention to a condition which denotes a hardware failure if the condition occurs frequently.

<unit mnemonic> I/O MEM PAR

The occurrence of this message denotes that a memory parity error has occurred during the transfer of data between an I/O channel and core memory. The MCP recognizes this error condition and rectifies the errors if possible. The primary purpose of this message is to draw attention to a condition which denotes a hardware failure if the condition occurs frequently.

■ # LOG HALF FULL

This message is typed if the log file SYSTEM/LOG is half full as a warning to the operator so that log information is not lost because of a log wraparound.

LOG WRAP AROUND

This message is typed if the MCP has to write on the beginning of the log file SYSTEM/LOG because of the fact that the log file has been filled and not reinitialized.

APPENDIX C (cont)
MESSAGES

LP BACKUP ON <unit mnemonic>

This message is provided to notify the operator that a print backup tape is on-line. (Operator action is not required unless it is desired to print the tape. If the tape is to be printed, a PB message must be entered.)

LP, PBT MT RQD <data file designator> <rdc> : <job specifier>

The occurrence of this message indicates that a program has need for a line printer or printer backup tape and neither is available. The situation denoted by this message is remedied if a line printer, backup tape, or scratch tape becomes available. The nature of the condition can be altered through use of the OU message.

LP RQD <data file designator> <rdc> : <job specifier>

The occurrence of this message indicates that a program has need for a line printer and such an I/O device is not currently available. The situation denoted by this message is remedied when a line printer becomes available; however, the OU message may be used to alter the nature of the condition.

-MAXN ARGMNT EXP: <job specifier> <terminal reference>

This message is typed if the argument to the EXP intrinsic exceeds 158 which causes the program to be terminated.

-MEMORY PARITY <job specifier> , <terminal reference>

The occurrence of this message denotes that a hardware failure has occurred which precluded further processing of the designated program. Consequently, processing of the program is discontinued.

MORE THAN 12000 CARDS IN <control card>

This message is typed when there are more than 12000 cards in a card deck which is being placed on the disk by LDCNTRL/DISK. This card deck is then completely removed from the disk.

MCP <version> . <release level> INCLUDES <MCP module list>

This message is typed in response to the keyboard input message WM.

APPENDIX C (cont)

MESSAGES

The message identifies the current MCP version, the patch level, and a list of the options under which the MCP has been compiled.

MT RQD <data file designator> <rdc> : <job specifier>

The occurrence of this message indicates that a program is in need of a scratch tape to use for a magnetic tape file.

-NEARLY SINGULAR MATRIX <job specifier> , <termination reference>

The occurrence of this message indicates that a BASIC program has attempted to invert a singular or nearly singular matrix. The program is discontinued.

-NEGTV ARGMNT LN <program specifier> <termination reference>

This message is typed upon the occurrence of a negative argument being passed to the LN intrinsic.

-NEGTV ARGMNT SQRT <program specifier> <termination reference>

This message is typed upon the occurrence of a negative argument being passed to the SQRT intrinsic.

NEW LOG FILE IS <m><d><c>/SYSLOG

This message occurs when the MCP initializes a new SYSTEM/LOG file, either in response to an LN keyboard message or automatically when the log is almost full. The new name given to the old SYSTEM/LOG file is <m><d><c>/SYSLOG where <m> is a 2-digit number representing the current month, <d> is a 2-digit number representing the day of the month, and <c> is a 2-digit number which is incremented each time the name-changing routine is invoked.

NEW PBT ON <unit>

This message is printed when a new printer backup tape is opened.

NEXT MCP WILL BE: <mfid>/<fid>

This message is typed in response to a valid CM message.

NO FILE <data file designator> <rdc> : <job specifier>

The occurrence of this message denotes that a program has need for

APPENDIX C (cont)
MESSAGES

an input file which is apparently not available. If the subject file is labeled, the situation denoted by this message may be remedied by making the file available.

If the file is labeled, the IL message must be used. If the file is a COBOL optional file, an OF message may be entered. If a COBOL program has read the final reel of a multi-reel unlabeled file, the FR message may be entered.

■ # NO FILE <library tape name> / FILE000:LIBMAIN/DISK = <mix index>

This message occurs when an attempt is made to LOAD files from a library tape which is not available to the system.

NO FIL ON DISK <data file designator> : <job specifier>

The occurrence of this message denotes that a program has need for a file it expected to find on disk. If the file noted in this message is made available on the disk so that the subject program can continue processing, the <mix index> OK message must be entered; again the MCP searches for the file to make it available to the program. If the file noted in the message cannot be made available, a DS message should be entered for the subject program.

■ NO MCP FILE <mfid> / <fid>

This message is typed in response to an invalid CM message.

<mix index> NO MEM

The occurrence of this message denotes that the MCP has made an attempt to obtain an area in core memory but was unable to do so. After not obtaining the area, the MCP allows other processing, if any, to take place; and subsequently makes periodic attempts to obtain the desired area. The NO MEM message is suppressed until five unsuccessful attempts have been made. If the area is ever obtained, the OK MEM message is typed. The <mix index> in this message denotes the program for which the area is to be obtained; 0 (zero) denotes the MCP. When the NO MEM message appears, it may or may not be followed by an OK MEM message. The system operator is required

APPENDIX C (cont)

MESSAGES

to determine actions subsequent to the NO MEM message; a HALT-LOAD operation may be required.

-NON-CONFORMAL ARRAYS <job specifier> , <terminal reference>

A BASIC program attempted to perform a matrix operation on two arrays which were dimensioned such that the result cannot be defined. The program is discontinued.

-NON-SQUARE MATRIX

A BASIC program attempted to either insert a non-square matrix or perform the IDN function on a non-square matrix. The program is discontinued.

NO SM STATIONS ON MIX = <mix index>

The mix SS message provides a means whereby a message can be sent to all "remote SPO" users of a particular mix who have requested mix messages via the SM message. The NO SM STATIONS ON MIX is typed if no users have requested messages.

<unit mnemonic> NOT A LIBRARY TAPE

A tape has been designated for a library load which is not in library-dump-tape format. If a nonmultifile tape has been specified, the tape is rewound and locked; and the proper tape is again requested. If a multifile tape that is not a library dump tape has been specified, LIBMAIN/DISK is terminated.

<file specifier> NOT DUMPED (DISK PARITY) <unit mnemonic>

This message indicates that a disk error occurred on the specified file. The user should attempt to dump the file and, if parity still exists, remove the file from the original dump deck.

<file specifier> NOT DUMPED (INV REC SIZE) <unit mnemonic>

This message indicates that some error condition occurred while attempting a library dump of the specified file. The user should dump that file to a different tape on a different unit. If the error occurs again, the file should be removed.

APPENDIX C (cont)

MESSAGES

⟨file specifier⟩ NOT DUMPED (INVALID USER)

This message indicates that the user code attached to the library maintenance dump operation is neither that of the creator of the file nor an entry in the security file attached to that file.

⟨file specifier⟩ NOT DUMPED (NOT ON DISK)

This message indicates that a file specified in a library maintenance dump operation is not present on disk.

⟨program-id⟩ NOT EXECUTABLE CODE

The occurrence of this message indicates that the MCP has performed a consistency check on a program and the program is not of executable form.

⟨file specifier⟩ NOT IN DIRECTORY

This message is typed if a control card references a file which is not in the disk directory.

⟨file specifier⟩ NOT LOADED (INV REC SIZE) ⟨unit mnemonic⟩

This message indicates that an error occurred when the library maintenance tape was created. It is suggested that the user move the tape to another unit and attempt to load the specified file.

⟨file specifier⟩ NOT LOADED (INVALID USER) ⟨unit mnemonic⟩

This message indicates that a file specified on a library load operation has a file security status conflict with a file of the same name already present on disk.

⟨file specifier⟩ NOT LOADED (NO USER DISK) ⟨unit mnemonic⟩

This message indicates that library maintenance is unable to load the specified file. LIBMAIN/DISK then attempts to load the next file.

⟨file specifier⟩ NOT LOADED (NOT ON TAPE) ⟨unit mnemonic⟩

The indicated file does not appear in the directory of a library maintenance tape.

APPENDIX C (cont)

MESSAGES

<file specifier> NOT LOADED (TAPE PARITY) <unit mnemonic>

The indicated file is not loaded because of inability to read the specified file. It is suggested that the user attempt to load the file from a different tape unit.

<unit mnemonic> NOT READY

The occurrence of this message denotes that the MCP or an object program has attempted to perform an I/O operation on the designated unit and has found the unit NOT READY.

<unit mnemonic> NOT READY EU NO. <digit> DA = nnnnnn MIX = <mix index>

The occurrence of this message denotes that the MCP or an object program has attempted to perform an I/O operation on the designated unit and has found the disk file electronics unit not ready.

<file specifier> NOT RESET (NOT ON DISK) or
<file specifier> NOT RESET (INVALID USER)

One of these messages is typed when an attempt is made to reset the access flag on a disk file by a control card, and is unsuccessful because of the reason indicated, provided the TYPE RSMSG option is set.

<file specifier> NOT SET (NOT ON DISK) or
<file specifier> NOT SET (INVALID USER)

One of these messages is typed when an attempt is made to set the access flag on a disk file by a control card, and is unsuccessful because of the reason indicated, provided the TYPE RSMSG option is set.

NO USER DISK

This message occurs if the MCP is requested to perform a library maintenance activity which requests an area on user disk, and no such area is available. If the condition indicated by this message should occur, a HALT-LOAD operation is required.

APPENDIX C (cont)

MESSAGES

NO USER DISK: <job specifier>

The occurrence of this message denotes that a program has attempted to obtain a file area on user disk, but an area of the required size is not available. If subsequent action is taken to make user disk available, the OK message must be entered to cause the MCP to again attempt to find the requested area. If user disk is not made available, a DS message should be entered for the program.

**NO USER DISK FOR RESERVE/DISK

This message is produced when the operator has entered MR through the SPO. The 2000 segments necessary for the RESERVE/DISK file are not available; the operator must enter a subsequent MR to create the file.

-NULL LIBRARY <file specifier>

This message indicates that a DUMP function proved to be vacuous. In other words, the item or items to be DUMPed do not exist on disk. If, for example, the operator requests that files A/= be DUMPed to a library tape called DEF and there are no files for which the first name is A, then the message is typed.

NULL REMOTE/LOG

This message is typed the first time the remote log is accessed and is not present.

NULL PBxxxx

This message is typed if an attempt is made to print a printer backup file while the PRNPBT/DISK routine is not in the directory, a printer is not available, or the file does not exist.

<mix index> OK MEM

This message may occur after a NO MEM message. The occurrence of this message denotes that the condition indicated by the NO MEM message no longer exists.

APPENDIX C (cont)

MESSAGES

-OPRTR DS-ED <job specifier> , <termination reference>

This message is typed after the system operator causes processing of a program to be discontinued through use of a DS message.

-OPRTR ES-ED <job specifier>

This message is typed after the operator causes a program to be eliminated from the schedule by an ES message.

OPRTR ST-ED <job specifier>

The occurrence of this message means that the job has been suspended in response to an ST keyboard input message. To resume processing of the program, the operator must use the OK message.

<unit mnemonic> OUT <data file designator> <rdc> : <job specifier>

This message is typed when a program opens a card, tape, or line printer file for output, providing the necessary options have been set. The message is typed for object program files if the TYPE OPEN option is set. The message is typed for compiler files if both the TYPE OPEN and TYPE CMPLFILE options are set.

-OUT OF DATA <job specifier> <terminal reference>

A BASIC program has attempted to read beyond the end of the data in its DATA statement. The program is discontinued.

<unit mnemonic> OUT PBTMCP BACKUP: <job specifier>

This message is typed when a scratch tape is initially selected and used for a printer backup tape, providing the necessary options have been set. The message is typed when an object program places the first file on a printer backup tape if both the TYPE OPEN and TYPE CMPLFILE options are set.

PARITY ERROR <job specifier>

A parity error has been encountered while printing a printer backup tape. The operator may respond with an OK, which continues the printing, or a DS.

PARITY ON <unit mnemonic>

The occurrence of this message means that the MCP has tried to read

APPENDIX C (cont)

MESSAGES

this tape and received an irrecoverable parity condition while reading the label information or scanning down a multifile reel.

<unit mnemonic> PARITY, RW/L

The occurrence of this message indicates that the MCP has attempted to read the designated magnetic tape unit, but has received a parity error condition and has consequently made the unit inaccessible.

The reason for the apparent parity condition might be that the tape unit has been set to the wrong density. If the subject unit is made ready again, either by placing the unit in LOCAL and then in REMOTE or through use of the RY message, the MCP makes another attempt to read the tape. Also, a PG message referencing the subject unit can be entered to purge the tape which makes it accessible.

-PAR NO LABEL <file designator> : <job specifier> , <termination reference>

The occurrence of this message indicates there has been an irrecoverable parity on the designated file and the object program has not specified any action for such a condition. Consequently, processing of the program is discontinued.

PBT MT RQD <data file designator> <rdc> : <job specifier>

The occurrence of this message indicates a program is in need of a scratch tape to use for a printer backup file. The situation denoted by this message is remedied when a scratch tape is made available. The nature of the condition can be altered through use of the OU message.

<unit mnemonic> PG-ED

This message is typed when a tape is purged either by a keyboard input message or a program.

<unit mnemonic> PRINT CHECK

This message is typed when a print check error has occurred during printing of a line on a line printer. This message is provided for the purpose of notifying the operator that the error has occurred;

APPENDIX C (cont)
MESSAGES

processing of the program using the line printer is continued as though the error has not occurred.

PP RQD <data field designator> <rdc> : <job specifier>

The occurrence of this message denotes that a program has need for a paper tape punch and no such I/O device is currently available.

<unit mnemonic> PUNCH CHECK

This message is typed when a punch check error has occurred during the punching of a card. This message is provided for the purpose of notifying the operator that the error has occurred; processing of the program using the card punch is continued as though the error has not occurred.

<unit mnemonic> READ CHECK

This message is typed when a read check occurs on a card reader. The operator must place the card in the card reader again. If the card is a badly worn card, it should be reproduced.

READ ERROR FOR {control card information}

The occurrence of this message denotes that a read error, probably irrecoverable parity, has occurred during the reading of a control deck for the disk. The control card which is printed out denotes the deck which is to be deleted because of this error. The following decks are still to be loaded.

<unit mnemonic> REL <data file designator> <rdc> : <job specifier>

This message is typed when a program closes a card, tape, or line printer file, providing the necessary options have been set. The message is typed for object program files if the TYPE CLOSE option is set. The message is typed for compiler files if both the TYPE CLOSE and TYPE CMPLFILE options are set.

REMOTE/LOG FULL

This message is typed when the log is full. Wraparound occurs the next time the log is accessed.

APPENDIX C (cont)
MESSAGES

REMOTE LOG ON DISK

This message is typed when the REMOTE/LOG has been placed on disk and initialized. If the file with <file identification prefix> REMOTE and <file identification> LOG is not on disk, 1000 segments are obtained for the REMOTE/LOG file and it is entered in the Disk Directory. The first 1000-ABRTLNGTH segments are reserved for log entries; the record capacity in logical records of this area equals 6^x (1000-ABRTLNGTH). The remaining segments are reserved for information pertinent to remote terminals currently attached to programs for abort logging if necessary. (An entry is made in this section of the file for each remote terminal attached to a job.) The maximum number of such entries is 3^x (ABRTLNGTH-1).

<file specifier> REMOVED

This message is typed after the MCP has performed an operation specified on a REMOVE control card.

-RER NO LABEL <file designator> : <job specifier> , <termination reference>

The occurrence of this message denotes that there is an R-format error on the designated input file and the object program has not specified any action for such a condition. Consequently, processing of the program is discontinued.

-RER NO LABEL <prog. id.> <termination reference>

This message is printed if an R-editing phrase error is detected on an array row read statement and an action label has not been specified.

RESERVE/DISK ALREADY PRESENT

This message is produced in response to the MR keyboard input message. The RESERVE/DISK file has already been created by a previous MR message.

RESERVE/DISK CREATED

This message indicates that the RESERVE/DISK file, consisting of

APPENDIX C (cont)

MESSAGES

one row of 2000 segments, has been created in response to the MR keyboard message.

<file specifier> RESET ACCESSD

This message is typed when the access flag of a disk file is reset by a control card, provided the TYPE RSMSG option is set.

<priority> : <job specifier> = <mix index> RESTARTED

The designated restart job has completed replacing core storage and now has a normal job structure; i.e., it has "restarted."

<priority> : <job specifier> = <mix> RESTART IS <state>

The <state> is either WAITING or MOVING. The operator requested <mix> WY before the job restarted. This response tells what RESTART is doing.

<unit mnemonic> RW/L

The occurrence of this message denotes that an operation has been performed to rewind the tape on the designated unit and to make the unit inaccessible. The unit may be made accessible again by placing it in LOCAL and then REMOTE, or through use of the RY message.

<unit mnemonic> RW/L <library tape name> (LIBRARY DUMP)

This message occurs after a library tape has been made through use of the DUMP card facility. The designated unit is the location of the newly created library tape, and the unit may be made accessible again by placing it in LOCAL and then in REMOTE, or through use of the RY message.

-SELECT ERROR <file designator> : <job specifier> , <termination reference>

The occurrence of this message denotes that an object program has performed an invalid operation on the designated file, e.g., rewinding a card reader. Consequently, processing of the program is discontinued.

APPENDIX C (cont)

MESSAGES

<file specifier> SET ACCESSD

This message is typed when the access flag of a disk file is set by a control card, providing the TYPE RSMMSG option is set.

SLATE OVRFLW

The occurrence of this message denotes that too many entries have been made in the SLATE, a table used by the control section of the MCP. If the condition indicated by this message should occur, a HALT-LOAD operation is required.

STA <integer> / <integer> MARKED NON-SPO

This is an output SPO message. The DC MCP has a means for "remembering" the locations of remote stations which have SPO capabilities. It is sometimes desired, however, to designate that certain stations should no longer be identified as "remote SPO's" (e.g., when an adapter which does not facilitate SPO facilities replaces a TWX adapter). In order that stations may be marked as "non-SPO stations," the RR keyboard input message is provided. If a message of the form RR <integer> / <integer> is entered, where the first <integer> specifies a terminal unit number and the second a buffer number, the MCP removes that station's identity as a "SPO station." Also, if the station is logged-in, the MCP logs it out.

-STACK OVRFLW <job specifier> , <termination reference>

The occurrence of this message denotes that the operations performed by an object program have caused its stack to overflow its limit. Consequently, processing of the program has been discontinued.

STATION <terminal unit> / <buffer> DISCONNECT AT <time>

This message is typed if a remote station becomes detached from the system without properly logging out.

STATION <terminal unit> / <buffer> LOGGED IN AT <time> BY <user code>

This message is typed when a user logs in from a remote station. If the file REMOTE/USERS is not present, <user code> is <empty>.

APPENDIX C (cont)

MESSAGES

STATION <terminal unit> / <buffer> LOGGED OUT AT <time>

This message is typed when a user logs out from a remote station.

<compiler name> / <program identifier> = <mix index> SYNTAX ERR

This message is typed when a compiler reaches End-of-Job and the program being compiled contains syntax errors, providing the TYPE EOJ option is set.

SYSTEM <system mnemonic> CLEARED

This message is typed in response to the CL <system mnemonic> message.

SYSTEM ERROR <data file designator> : <job specifier>

Because the MCP and INTRINSICS no longer reside in a reserved location on disk, there is now a new class of system files. These files are not reserved by name but are marked as system files in the Disk Directory. In addition to having library maintenance performed on these files disallowed, the files may not be accessed in such a manner that their information content could be changed. If an attempt is made to improperly access one of these files, the job is suspended until the operator intervenes. The operator may then either DS the job or take action to change the file in question to a non-system file and then type <mix index> OK. (For instance, if a System 1 program attempts to change the INTRINSICS file currently in use by System 2, the INTRINSICS on System 2 would need to be changed before the System 1 program could continue.)

<unit mnemonic> TAPE MK, RW/L

The occurrence of this message indicates that the MCP has attempted to read the designated magnetic tape unit, has found the first word of information to be a tape mark, and has consequently made the unit inaccessible. The reason for the apparent tape mark condition may be that the tape unit has been set to the wrong density. If the subject unit is made ready again, either by placing the unit in LOCAL and then in REMOTE or through use of the RY message, the MCP again attempts to read the tape. Also, a PG message referencing

APPENDIX C (cont)

MESSAGES

the subject unit can be entered, and the tape is purged and made accessible.

TR PLEASE

This message is typed at HALT-LOAD time if the TYPE TIME option is set. The system operator is required to enter a TR message before processing can continue.

-TYPE MISMATCH READ STMT <job specifier> , <terminal reference>

A BASIC program has tried to READ data that is not in the proper form to satisfy the input list. The program is discontinued.

UNEXP IO ERR

The occurrence of this message denotes that the MCP has encountered an unexplained I/O error that cannot be directly associated with a particular program. If this error should occur, a HALT-LOAD operation is required.

-UNEXP I-O ERROR, <unit mnemonic>, LIB MAINT ABORTED

The occurrence of this message indicates that a library load or dump has been aborted because of an unexpected I/O error. If this occurs during a load operation, the file currently being loaded is removed from the directory.

UNEXP I-O ERROR ON <unit> : RESULT = <error field>; MASK = <mask>.

where:

<unit> ::= the unit mnemonic (CRA, MTC, DKA, etc.).

<error field> ::= result descriptor error field ([26:7]).

<mask> ::= mask specified by the calling procedure. A bit ON in the mask indicates a condition to be handled by the calling procedure.

The message indicates that the MCP has performed an I/O operation which caused an unexpected I/O error. If the I/O operation is directly related to a particular object program, processing of that program is discontinued. Also, the MCP types the error field, the unit mnemonic, and the error mask field as additional information.

APPENDIX C (cont)

MESSAGES

<file name> UNOPENED

This message states that the I/O area was referenced while the file was not open.

<unit mnemonic> WRITE LOCK

The occurrence of this message denotes that a program has attempted to write on a magnetic tape with no write ring, or on a disk or drum which has been locked out through use of hardware lockout switches. Consequently, processing of the program using the unit has been discontinued.

<unit mnemonic> WR PARITY

The occurrence of this message denotes that an irrecoverable write parity has occurred on the designated unit. Consequently, processing of the program using the unit has been discontinued.

--WRONG FILE <data file designator> <rdc> : <job specifier>

Reopening files on a restart attempt involves checking for compatibility with those in use at breakout. This message indicates that the designated file is not sufficiently compatible; the next message hints why. The operator may reply with an OK, WY, or DS. OK initiates a recheck.

--WRONG <hint>, ..., <hint>

This message pertains to BREAKOUT RESTART and the hints include the following:

- a. WRITE STATE. The write ring of the file is in the wrong place (in the box or on the tape).
- b. LABEL. The file lacks (needs) a label.
- c. TYPE. A file on disk (tape, line printer) at break must be on disk (tape, line printer) at restart. For example, an LP file is broken and the operator tries reopening it as a PBD.

APPENDIX C (cont)

MESSAGES

- d. ROWS USED. Disk files have up to 20 rows. The one found does not have the right one.
- e. NO. OF ROWS. The disk file found has the wrong number of rows allowed.
- f. EOF. It is too short.
- g. ROW LENGTH. Its rows are the wrong size.
- h. FORMAT. If it is a disk file, its blocking or record length is wrong. If it is a tape file, it is found beyond where it was in breakout.
- i. SECURITY. A security error occurred.
- j. NO DUMP. The tape file lacks a break file copy.

ZIP ERROR - IGNORED

This message is typed if a program performs a generalized ZIP statement but provides control information containing an error. Occurrence of this message signifies that the error is present and that all control information following and including the error is ignored.

-ZERO ARGMNT LN <program specifier> <termination reference>

This message is typed upon the occurrence of an argument of zero being passed to the LN intrinsic.

KEYBOARD INPUT MESSAGES.

Operator messages are defined as messages with a free-field format which the operator can supply the MCP via the console keyboard. In keeping with the concept of permitting the system to perform the control functions, the operator messages are primarily restricted to those actions that facilitate processing. The messages are not intended to provide detailed information about individual programs, e.g., the settings for specific registers or the contents of designated memory locations.

APPENDIX C (cont)

MESSAGES

To enter information from the keyboard, the operator must first press the INPUT REQUEST key. The READY indicator on the supervisory printer is turned on. At this time, the operator can enter his message. When he has finished keying in the message, he presses the END OF INPUT key. This key causes a group mark to be inserted immediately after the last character entered and signals the MCP.

If the operator attempts to introduce a message that is not acceptable, the MCP ignores it and notifies the operator that he has keyed in an invalid entry.

The following list presents the allowable operator input messages.

THE AX MESSAGE

The AX message allows the console operator to communicate with the object program through the SPO in response to an ACCEPT message.

All characters following the AX, including blanks, are available to the object program. Following the last character typed is a group mark.

The AX message has the following format:

⟨mix index⟩ AX ⟨message data⟩

THE BK MESSAGE

This message performs the equivalent of the break key function for a SPO console or the SPO. If ⟨mix index⟩ is specified, messages which have been queued up for the SPO by a particular program in the mix are discarded.

The BK message has the following format:

BK or ⟨mix index⟩ BK

Examples

BK

3BK

APPENDIX C (cont)
MESSAGES

THE BS MESSAGE

The BS message sets the station indicated by <tu>/<buff> as a SPO console. (Refer to the description of SPO consoles.)

The BS message has the following format:

BS <tu>/<buff>

Example

BS 1/8

THE BS SPO MESSAGE

This message causes SPO output to be printed on the SPO.

The BS SPO message has the following format:

BS SPO

Example

BS SPO

THE CC MESSAGE -- ? MESSAGE

The CC message allows the system operator to supply control information to the MCP via the console typewriter. The information following the letters CC in the CC message is recognized in the same fashion as the information following the character ? on control cards and program-parameter cards.

The character ? can be used in lieu of the characters CC in the CC message, if desired.

When a CC message is entered and the END OF INPUT switch is pressed, the typewriter becomes READY again unless the CC message contains END card information. Consequently, the last CC message must always be an END card message.

APPENDIX C (cont)

MESSAGES

The term <control information> used below is defined as any information defined valid for use on control cards or program-parameter cards.

The CC message may have either of the two following formats:

CC <control information>

or

? <control information>

Examples

CC EXECUTE C/P; END

CC EXECUTE C/P

CC END

? COMPILE "00180" BY IRP WITH ALGOL

? ALGOL FILE CARD = IRACARD

? END

? COMPILE A/B; ALGOL FILE CARD = "OXXXXXX"; END.

THE CD MESSAGE

The CD message causes the MCP to type the name and first card image of each pseudo card deck that has been placed on the disk by the LDCNTRL/DISK Program. The CD message can also specify the system for which decks are to be printed. The messages are as follows:

- a. CDSYA
- b. CDSYB
- c. CDSYC
- d. CDSYD
- e. CDALL

If CD alone is typed, the system into which the message is entered is implied. If there are no pseudo card decks on disk, the following is typed:

NO DECKS ON DISK

APPENDIX C (cont)
MESSAGES

The CD message has the following format:

CD

THE CI MESSAGE

The CI message causes the MCP to forget the current intrinsic file. If, at a time when an intrinsic file is already on disk, a new intrinsic file has been created, and the operator wishes to ~~remove~~^{stop using} the old file and use the new one, the CI (for Change Intrinsic) message may be used. If the designated file is found, the MCP waits until the only jobs being processed are either LDCNTRL/DISK, PRNPBT/DISK, or LIBMAIN/DISK, or a combination of these jobs, and then performs the change. The change is also performed if nothing is in the mix.

All systems may use the same intrinsic file, separate intrinsic files, or any combination thereof. The name of the intrinsic file is no longer changed to INTRNSC/DISK after the CI message is entered; therefore, the name INTRNSC/DISK no longer has any special meaning.

The CI message has the following format:

CI <file identification prefix> <separator>
<file identification>

Example

CI INT/DISK

THE CL <unit mnemonic> MESSAGE

The CL message discontinues the job using the specified unit. If the job is using the specified unit, the label table entry for the unit is marked as not in use and the unit is made ready.

The CL message has the following format:

CL <unit mnemonic>

APPENDIX C (cont)

MESSAGES

Example

CL MTA

THE CL <system mnemonic> MESSAGE

Operationally, the Sharedisk System is very similar to the present batch system. If a system within the network is HALT/LOADed, it first determines if there are other systems also in operation. If so, the HALT/LOADed system closes all disk files that were open and returns all user disk that was in use (without disturbing the other systems). If, on the other hand, the HALT/LOADed system is the only one presently running, it takes the same action that a non-shared disk system takes at HALT/LOAD (rebuilds the Disk Directory, etc.). If a system ceases to function and cannot be restarted because of a hardware failure, the message CL <system mnemonic> should be typed into a functioning system. This causes all addresses in File Protect Memory that were locked by the disabled system to be unlocked, closes all files that were open, and returns all user disk that was in use. After this process is completed, the message #SYSTEM n CLEARED is typed; the system then performs the following tasks:

- a. Removes all contention bits set by SYN SY<n>.
- b. Unlocks all addresses locked by SYN SY<n>.
- c. Returns all disk space in the Scratch Directory of the system that is being cleared.
- d. Removes all files being loaded by the system that is being cleared.
- e. Closes all files that the offending system has opened.
- f. Removes all entries in the hold-list made by the system being cleared.
- g. Wakes up all processes in other systems that are waiting for a file that is in use (by any system).

APPENDIX C (cont)

MESSAGES

NOTE

It is recommended that the CL message be used when a system ceases to function for any reason. It has been found that the performance of the other systems within the network may be seriously reduced while a HALT/LOAD is being performed unless the disabled system is first cleared via the CL message.

THE CM MESSAGE

The MCP may be located anywhere on disk in the same fashion as any other file. This makes it possible for all systems to use MCP's with the same level with different options included (e.g., data communications, debugging) if so desired. The MCP may be loaded with the Tape to Disk Loader or, if the MCP is already on disk, the Disk to Disk Loader. If it is desired to change MCP's, the keyboard message CM <mid>/<fid> is entered. If that file is in the Disk Directory, the message NEXT MCP WILL BE: <mid>/<fid> is typed and that MCP is used as soon as a HALT/LOAD occurs. If that file is not in the Disk Directory, the message NO MCP FILE <mid>/<fid> is printed and no further action takes place. When changing from a non-SHAREDISK MCP to a SHAREDISK MCP, the COOL START Program must be run prior to HALT/LOADing. Similarly, while running on the SHAREDISK MCP, any attempt to CM a non-SHAREDISK MCP will be unsuccessful.

THE CT MESSAGE

The CT message is used to change the time limits for a job. For either I/O or processor time limits, if the input is a non-zero integer, the time limit is changed to these new values.

The CT message has the following format:

<mix index> CT <processor part> <I/O part>

APPENDIX C (cont)
MESSAGES

THE CU MESSAGE

The CU message allows the console operator to determine the core usage for a single program or all programs in the mix.

The CU message has the format:

<mix index> CU

or

CU

THE DB MESSAGE

The DB message is operational for those MCP's compiled with DEBUGGING set TRUE. This message places calls on MCP procedures which, through a special symbol/field-oriented language, allow the reading and writing of any area on disk. The DB message is used to initiate a debugging session; the semicolon terminates it.

NOTE

Because control is not maintained by the MCP when reading and writing a word to an address, the integrity of the system is the responsibility of those utilizing this function.

THE DD MESSAGE

The DD message is operational for those MCP's compiled with DEBUGGING set TRUE. This message places calls on MCP procedures which, through a special symbol/field-oriented language, allow the reading and writing of any area in core. The DD message is used to initiate the debugging session; the semicolon terminates it.

NOTE

Because control is not maintained by the MCP when reading and writing a word to an address, the integrity of

APPENDIX C (cont)

MESSAGES

NOTE (cont)

the system is the responsibility of those utilizing this function.

THE DP MESSAGE

The DP message is operational for those MCP's compiled with DEBUGGING and/or DUMP set TRUE. This message allows an octal core dump to a line printer or a magnetic tape unit.

The DP message has the following format:

DP<unit>

where <unit> is LP for line printer and MT for magnetic tape.

THE DS MESSAGE

The DS message allows the system operator to cause a program to be terminated.

There are two forms of the DS message. One form of the message requires that the program to be terminated be identified through use of a <mix index>* term; the other message requires that the program be identified through use of a <program specifier>.

If two or more programs in a mix have the same <program name> and a message using a <program specifier> is entered, the MCP arbitrarily terminates the program, with the name specified, that has the lowest <mix index>. Consequently, if a situation such as noted occurs, the DS message which identifies the program through use of the <mix index> term is used.

The DS message may have either of the two following formats:

<mix index> DS

*The term <mix index> is an <integer> that represents the mix index that the MCP has assigned to a particular program in the mix; i.e., a program that is currently in process.

APPENDIX C (cont)
MESSAGES

or

DS <program specifier>

Examples

2 DS
DS ALGOL/"00180"

THE DT MESSAGE

The DT message allows the system operator to change the value of the current date word used by the MCP.

The DT message requires the use of three <integer>s, the first two of which must be followed by the character /. The first <integer> is recognized as the number of the month of the year, the second <integer> is recognized as the day of the month, and the third <integer> is recognized as the last two digits of the year.

The DT message has the following format:

DT <integer> / <integer> / <integer>

Example

DT 1/1/67

THE ED MESSAGE

The ED message can be used to eliminate a pseudo card deck which is contained in a pseudo card reader if the reader is not in use.

The ED message may have one of the following formats:

ED CDA
ED CDB
ED CDC
ED CDD

APPENDIX C (cont)
MESSAGES

THE EI MESSAGE

The EI message responds with the message EIO and performs no useful function.

THE ES MESSAGE

This message terminates a program which is still in the schedule. This cannot be done with a DS message because the program is in the schedule and not in the mix. The program may be eliminated from the schedule by typing in <schedule index> ES (meaning to eliminate from schedule). This causes the program to be loaded into the mix and DSed before any of its statements are executed.

THE EX MESSAGE

The EX message types out all expired disk file names.

Examples

```
EX ALGOL/=
EX =/DISK
EX
```

THE FM MESSAGE

The FM message must be entered in response to a # FM RQD message. The <mix index> in the message must agree with the <mix index> in the # FM RQD message, and the <unit mnemonic> must designate the unit to be used for the subject file.

The FM message has the following format:

```
<mix index> FM <unit mnemonic>
```

Example

```
1 FM LPB
```

THE FR MESSAGE

The FR message allows the system operator to specify that the input

APPENDIX C (cont)

MESSAGES

reel, the reading of which has just been completed, is the final reel of an unlabeled file.

The FR message has the following format:

<mix index> FR

Example

3 FR

THE IL MESSAGE

The IL message is used in response to a NOFILE or DUPFIL message and allows the system operator to designate the unit on which a particular input file is located. The unit designated in the IL message may denote the location of a nonstandard file (a file with no standard label) or a standard file (a labeled file). In either case, the file on the unit designated in the IL message is assumed to be the file required in the related NOFILE or DUPFIL message.

A <mix index> term must be used with the IL message since, during multiprocessing, more than one NOFILE or DUPFIL message may be in effect at the same time.

The IL message must have the following format:

<mix index> IL <unit mnemonic>

Example

1 IL MTF

THE IN MESSAGE

The IN message allows the system operator to insert an <unsigned integer> into the Program Reference Table (PRT) of the program specified by the <mix index> at the relative location specified by the octal <index> unless the specified PRT cell contains a descriptor, or the <index> is less than 25 <octal> or out of the PRT bound.

APPENDIX C (cont)
MESSAGES

The IN message has the following format:

<mix index> IN <index> = <unsigned integer>

Example

2 IN 32 = 563

THE LD MESSAGE

The LD message causes the LDCNTRL/DISK Program to be called into core for execution. The LDCNTRL/DISK Program then searches for a tape or card file with the <multiple file identification>

CONTROL

and the <file identification>

DECK

Then, if the message entered is

LD DK

the file CONTROL/DECK is placed on disk in such a fashion that the MCP can read the file as a pseudo card deck. If the message entered is

LD MT

the file CONTROL/DECK is placed on a magnetic tape.

The LD message may have either of the following formats:

LD DK

or

LD MT

Examples

LD DK

LD MT

APPENDIX C (cont)
MESSAGES

THE LN MESSAGE

The LN message activates an MCP routine which changes the name of the SYSTEM/LOG and initializes a new SYSTEM/LOG file. The new name given to the old SYSTEM/LOG is $\langle m \rangle \langle d \rangle \langle c \rangle / \text{SYSLOG}$; where $\langle m \rangle$ is a 2-digit number representing the current month, $\langle d \rangle$ is a 2-digit number representing the day of the month, and $\langle c \rangle$ is a 3-digit number that is incremented each time the name-changing routine is invoked. The LN message has the following format:

LN

Example

LN

A keyboard message, which gives the new name, is written after the name has been changed.

Example

**** NEW LOG FILE IS 1230007/SYSLOG

In addition to being initiated by the LN message, the name-changing routine is automatically initiated when the log is almost full.

THE LNDK MESSAGE

The LNDK message causes all files on disk to be logged out and resets their creation date and time to current values.

THE LR MESSAGE

The LR message causes the library program with the $\langle \text{program identifier} \rangle$

LOGOUTR

and the $\langle \text{program identifier suffix} \rangle$

DISK

to be scheduled for execution.

APPENDIX C (cont)
MESSAGES

The LR message has the following format:

LR

Example

LR

THE MR MESSAGE

The MR message is used to create a file labeled RESERVE/DISK consisting of one row of 2000 segments in length. This file is used to create a buffer in cases of NO USER DISK. When a NO USER DISK occurs, RESERVE/DISK is automatically removed if the requested space is less than that in RESERVE/DISK. The removal of the file is indicated by the NO USER DISK message followed by

** RESERVE/DISK REMOVED

Once the RESERVE/DISK file has been removed, the operator must enter MR to re-create the file once space is available. The possible responses to the MR message are:

RESERVE/DISK CREATED

RESERVE/DISK ALREADY PRESENT

** NO USER DISK FOR RESERVE/DISK

Example

MR

THE MX MESSAGE

The MX message allows the system operator to request that the MCP type a list of <program specifier>s denoting the programs in the mix; the priority and <mix index> for each program is also listed.*

* It should be noted that the maximum number of programs allowed in the mix is determined by two parameters which are DEFINED in the MCP. The two parameters are MIXMAX and JOBNUMAX, where MIXMAX may be DEFINED as an integer from 1 through 29 and JOBNUMAX must be DEFINED as an even integer with a value of $20 \times \text{MIXMAX} + 30$.

APPENDIX C (cont)

MESSAGES

Specifically, each item in the list typed by the MCP in response to the MX message has the following format:

⟨priority⟩ : ⟨program specifier⟩ = ⟨mix index⟩

If there is nothing in the mix, the following message is typed:

NULL MIX

The MX message has the following format:

MX

THE OF MESSAGE

The OF message allows the system operator to specify that a file requested for a COBOL program is optional so that the specified program can proceed without it.

The OF message has the following format:

⟨mix index⟩ OF

Example

1 OF

THE OK MESSAGE

The OK message causes the MCP to resume processing of a program which has been temporarily suspended because of the condition designated by the # DUP LIBRARY message, the NO USER DISK message, the NO FILE ON DISK message, or the #OPRTR ST-ED message.

The OK message has the following format:

⟨mix index⟩ OK

Example

1 OK

APPENDIX C (cont)

MESSAGES

THE OL MESSAGE

The OL message allows the system operator to request that the MCP type information pertaining to labels of files on I/O units.

The OL message has many formats. One format specifies that a specific <unit mnemonic> may be entered. The other formats require 2-letter codes which specify a type of I/O unit. The codes and the I/O units they represent are as follows:

<u>Code</u>	<u>I/O Unit</u>
CD	Pseudo card reader
CP	Card punch
CR	Card reader
LP	Line printer
MT	Magnetic tape
PP	Paper tape punch
PR	Paper tape reader

If an OL message specifying a specific <unit mnemonic> is entered, the response message has one of the following formats, whichever is relevant.

<unit mnemonic> <physical reel number> IN USE BY <program specifier> : <multiple file identification> <file identification> <rdc>

<unit mnemonic> <physical reel number> LABELED <multiple file identification> <file identification> <rdc>

<unit mnemonic> <physical reel number> SCRATCH

<unit mnemonic> <physical reel number> UNLABELED

<unit mnemonic> NOT READY

If an OL message specifying a type of I/O unit is entered, and if a unit of the specified type is in use and/or labeled, the response message has one of the following formats, whichever is relevant.

APPENDIX C (cont)

MESSAGES

<unit mnemonic> <physical reel number> IN USE BY <program specifier> : <multiple file identification> <file identification> <rdc>

<unit mnemonic> <physical reel number> LABELED <multiple file identification> <file identification> <rdc>

<unit mnemonic> UNLABELED

If an OL message specifying a type of I/O unit is entered, and no unit of that type is in use and/or labeled, the following message is typed:

NULL <unit mnemonic> TABLE

The OL message may have one of the following formats:

OL <unit mnemonic>
OL CD
OL CP
OL CR
OL LP
OL MT
OL PP
OL PR

Examples

OL MTA
OL CR
OL MT
OL MTX (where X calls for a list of all scratch tapes)

THE OT MESSAGE

The OT message allows the system operator to request the MCP to type out the value of a cell in the Program Reference Table (PRT) of a

APPENDIX C (cont)

MESSAGES

program. The program is specified by the \langle mix index \rangle and the cell by the octal \langle index \rangle . The typed MCP message has the following format:

\langle job specifier \rangle : R+ \langle index \rangle = \langle PRT data \rangle

The value of \langle PRT data \rangle is expressed as an octal number for a descriptor, or an integer of up to eight digits for an operand.

The OT message has the following format:

\langle mix index \rangle OT \langle index \rangle

Example

2 OT 32

THE OU MESSAGE

The OU message allows the system operator to designate the output media option for a line printer file if a # LP RQD, a # LP PBT MT RQD, or a # PBT MT RQD message has been typed which references the job that uses the file.

The OU LP form of this message specifies that the subject line printer file must be produced as output on a line printer.

The OU MT form of this message specifies that the subject line printer file must be produced as output on a printer backup tape.

The OU DK form of this message specifies that the subject line printer file must be produced as output on printer backup disk.

The OU form of this message specifies that the subject line printer file may be produced as output either on a line printer or a printer backup tape. The OU message may have any one of the following formats:

\langle mix index \rangle OU LP

\langle mix index \rangle OU MT

\langle mix index \rangle OU

\langle mix index \rangle OU DK

APPENDIX C (cont)

MESSAGES

Examples

2 OU LP

1 OU

4 OU

THE PB MESSAGE

The PB message allows the system operator to specify that a printer backup file on a particular unit is to be printed. Any printer backup disk file may be printed on any (or all) system. If the file is being printed simultaneously on two or more systems (not considered a normal activity), the last system to finish printing removes the file from the Disk Directory. The PB command remains unchanged. If a specified tape is not a printer backup tape, the following message is typed:

NOT PRINTER BACKUP TAPE

The PB message has the following formats:

PB <unit mnemonic>

PB <PBD number>

The term <PBD number> may be up to four digits and is the nnnn part of the PBD file name.

Example

PB MTN

THE PD MESSAGE

The PD message allows the system operator to request that the MCP type information pertaining to what files are listed in the Disk Directory. The formats of the PD message are shown below. The action caused by the PD message depends upon the format of the message. Specifically, the actions caused by the PD message are as follows.

If a message of the form

APPENDIX C (cont)
MESSAGES

PD =/=

is entered, a list containing a <file specifier> for each file in the Disk Directory is typed.

If a message of the form

PD <file specifier>

is entered and the file designated in the message is in the Disk Directory, the <file specifier> for the file is typed. If the file designated in the message is not in the Disk Directory, the message

NULL PD <file specifier>

is typed.

If a message of the form

=/<file identification>

or

=/<program identification suffix>

is entered, a list of all files in the Disk Directory which have the designated <file identification> or <program identification suffix>, if any, is typed. If no such files are in the Disk Directory, one of the following messages is typed:

NULL PD<file identification prefix>/=

NULL PD<file identification prefix>

NULL PD<program identification>/=

NULL PD<program identification>

The PD message may have any one of the following formats:

PD =/=

PD <file specifier>

PD =/<file identification>

PD =/<program identification suffix>

APPENDIX C (cont)

MESSAGES

```
PD <file identification prefix>/=  
PD <file identification prefix>  
PD <program identification>/=  
PD <program identification>
```

Examples

```
PD =/  
PD ALGOL/DISK  
PD =/PARTS  
PD =/DISK  
PD PERSNEL/=   
PD PERSNEL  
PD ALGOL/=   
PD ALGOL
```

THE PG MESSAGE

The PG message allows the system operator to purge a magnetic tape on a unit that is in REMOTE mode, in WRITE RING status, and not in use. One form of the PG message allows a physical reel number to be inserted into the scratch label written on the tape. The physical reel number can be altered only by using the form of the PG message that specifies the number; it is not altered by the normal PG message. The format is:

```
PG <magnetic tape unit mnemonic> <physical reel number specifier>  
  
<physical reel number specifier> ::= -<5 digits>
```

If the <physical reel number specifier> format is used, there must not be any space between the last character of the <magnetic tape unit mnemonic>, the dash, and the <5 digits>.

Examples

```
PG MTA  
PG MTA-12345
```

APPENDIX C (cont)

MESSAGES

THE PR MESSAGE

The PR message provides a means whereby the system operator can specify the priority to be assigned a program currently in the mix. The priority to be assigned is specified by the term \langle priority \rangle ; the program to which the priority is to be assigned is specified by the \langle mix index \rangle . The term \langle priority \rangle must be an \langle integer \rangle .

The PR message has the following format:

```
 $\langle$ mix index $\rangle$  PR  $\langle$ priority $\rangle$ 
```

Example

```
4PR7
```

THE PS MESSAGE

The PS message can be used to alter the priority of a program in the schedule. The priority to be assigned is specified by the term \langle priority \rangle ; the program in the schedule to which the priority is to be assigned is specified by the \langle schedule index \rangle . Both terms are integers.

The PS message has the following format:

```
 $\langle$ schedule index $\rangle$  PS =  $\langle$ priority $\rangle$ 
```

Example

```
5PS = 3
```

THE PT MESSAGE

The PT message causes the declared disk file TRACE/AREA to be printed. This file contains trace output data when the MCP is compiled with DEBUGGING set TRUE and when calls are made on the TRACE procedure.

THE QT MESSAGE

The QT message is used to cause PRNPBT/DISK or LIBMAIN/DISK to skip a file. In response to this message, the following actions for PRNPBT/DISK files are:

APPENDIX C (cont)

MESSAGES

- a. Printing of the current file is discontinued.
- b. Printing resumes with the first record of the succeeding file.
- c. If the file being printed is a printer backup disk file, printing is discontinued and the appropriate copy of PRNPBT/DISK terminates. The PBD file is not removed from the Disk Directory.

The QT message may have either of two formats:

⟨mix index⟩ QT

or

QT ⟨unit mnemonic⟩

Example

1 QT

QT MTA

THE QV MESSAGE

The Q-second timeout facility provides a means for causing the MCP to clear read-ready conditions from the terminal unit buffers of remote stations which have SPO capabilities in cases where an object program (to which the station is assigned) does not read the input within a given number of seconds; i.e., within Q seconds. The value of Q may be specified through use of a QV keyboard input message of the form:

QV = ⟨integer⟩

In response to this message, the MCP sets Q to the value specified and types a SPO message of the form:

REMOTE SPO Q VALUE = ⟨integer⟩ SECS

Also, if a keyboard message of the form

QV

APPENDIX C (cont)
MESSAGES

is entered, the MCP responds with a message, as shown above, which specifies the current value of Q.

THE RD MESSAGE

The RD message may be used to remove pseudo card decks from disk which were placed on disk by the LDCNTRL/DISK Program.

Pseudo card decks are identified by names having the following format:

<integer>

and the term <pseudo card deck list> is defined as:

= | # <integer> | # <integer>, ..., # <integer>

If the keyboard message RD= is entered, only those decks that were loaded on the system into which the message is entered are removed. If the message RD#nnnn is entered, the specified deck is removed regardless of which system loaded it, providing the deck is not in use. The RD message has the following format:

RD <pseudo card deck list>

Examples

RD #0072
RD #0072, #6328
RD =

THE RM MESSAGE

The RM message can be used in response to a # DUP LIBRARY message. The RM message causes the file on disk (with a name identical to the file created by the program specified in the # DUP LIBRARY message) to be removed, and then causes the subject program to resume processing.

The RM message has the following format:

<mix index> RM

APPENDIX C (cont)
MESSAGES

Example

1 RM

THE RN MESSAGE

The RN message is used to specify the number of pseudo card readers to be used. There are four pseudo card readers in the Time Sharing MCP and 32 in the Standard MCP. At HALT/LOAD time, the number of pseudo card readers specified to be used is zero.

An RN message may be entered at any time. If an RN message specifies that more pseudo card readers are to be used than are currently being used, the MCP searches for pseudo card decks on disk and makes use of as many of the specified pseudo card readers as possible. If an RN message specifies that fewer pseudo card readers are to be used than are currently being used, a sufficient number of the pseudo readers are "turned off" as soon as the readers complete handling of the pseudo card deck in process, if any.

All control decks are identified with the system that loads them, and run on that system unless special action is taken. A variation of the RN message is RN#nnnn where nnnn is the number of the control deck on disk. Providing the specified deck is not already in use, typing in this message places the deck in a pseudo reader regardless of which system loads it. If the deck is in use, the message DECK#nnnn IN USE BY SYSTEM n is typed.

The RN message may have one of the following formats:

RN
RN <digit>
RN#nnnn

If a <digit> is not entered, 1 is assumed.

APPENDIX C (cont)
MESSAGES

Examples

RN
RN 0
RN 1
RN 2
RN 3
RN 4
RN #1234

THE RR MESSAGE

The DC MCP has a means for "remembering" the locations of remote stations which have SPO capabilities. It is sometimes desired, however, to designate that certain stations should no longer be identified as "remote SPO's;" e.g., when an adapter which does not facilitate SPO facilities replaces a TWX adapter. In order that stations may be marked as "non-SPO stations," the RR (Remove Remote) keyboard input message is provided. If a message of the form

RR <integer> / <integer>

is entered (where the first <integer> specifies a terminal unit number and the second a buffer number), the MCP removes the identity of that station as a "SPO station." Also, if the station is logged in, the MCP logs it out.

THE RS MESSAGE

This message allows the operator to add a break file to the Directory, the break file having been previously copied onto an output tape of a COBOL job.

The break file, loaded onto disk from the specified tape unit by the RS message, may then be executed or run to initiate a restart.

The RS message has the following format:

RS <unit>

APPENDIX C (cont)
MESSAGES

The responses are:

- a. RS <unit> INV KBD
The unit is not a tape.
- b. <unit> <note>
The note is either NOT READY, IN USE, SCRATCH, WRITE LOCK, or NO DUMP. The unit must be an available, labeled, write-enabled tape with a break file copy.
- c. . <program name> / BREAK <break number> NOT ADDED . DUP
LIB RS <unit>
There is already a disk file with the names mentioned. The break file on the unit is not loaded.
- d. <unit> ERROR IN DUMP
The tape/disk break file copy has gone awry. The break file is not loaded.
- e. <program name> / BREAK <number> ADDED . TAPE POSITIONED :
<unit>
The RS <unit> is successful. The unit is left positioned and marked for the pending restart of the loaded break file.

Example

RS MTA

THE RW MESSAGE

The RW message allows the system operator to cause a rewind-and-lock action to be performed on a magnetic tape file that is not in use.

The RW message has the following format:

RW <unit mnemonic>

Example

RW MTE

APPENDIX C (cont)

MESSAGES

THE RY MESSAGE

The RY message allows the system operator to cause, by entering a keyboard message, an effect analogous to the effect caused by placing a magnetic tape unit in LOCAL and then REMOTE. That is, if the designated unit is not in use and in REMOTE, the MCP attempts to read a file label.

The RY message causes locked files to be made accessible and causes label cards (or DATA cards), which have been read but not referenced, to be ignored.

The RY message has the following format:

```
RY <unit mnemonic> <blank> | <unit mnemonic> <unit mnemonic>
```

Examples

```
RY MTC
```

```
RY CRACRBMTA
```

THE SC MESSAGE

The SC message types the SPO consoles.

The SC message has the following format:

```
SC
```

THE SI MESSAGE

When the MCP is compiled with STATISTICS set TRUE, the SI message allows the time-transfer value to be changed. The time-transfer value is used to calculate when to transfer the SYSTEM<system mnemonic>/STATS File into the SYSTAT<system mnemonic>/DISK File.

The SI message has the following format:

```
SI<integer in minutes>
```

The value of <integer in minutes> is initially 30.

APPENDIX C (cont)

MESSAGES

THE SL MESSAGE

The SL message allows the operator to save the Pseudo Statistics (MICRO) Log File. (Refer to Statistics Log in section 5.)

The SL message has the following format:

SL

THE SO, RO, and TO OPTION MESSAGES

The MCP provides a number of features that are optional. That is, if a particular option is set, the MCP uses the respective feature; if the option is reset (i.e., not set), the feature is not used. (Refer to page 3-26 for an explanation of options.)

The SO message allows the system operator to set options.

The RO message allows the system operator to reset options.

The TO message allows the system operator to request that the MCP type a message which lists the options and their settings.

Each optional feature provided by the MCP may be referenced either mnemonically through use of an <option mnemonic> or numerically through use of an <option numeric code>.

An <option mnemonic> is defined as one of the following:

USE DRA

USE DRB

TYPE BOJ

TYPE EOJ

TYPE OPEN

USE TERMNATE

TYPE DATE

TYPE TIME

USE ONEBREAK

USE AUTOPRNT

APPENDIX C (cont)
MESSAGES

USE CLEARWRS
TYPE DISCONDC
TYPE CmplFILE
TYPE CLOSE
USE ERRORMSG
USE RET
USE LIBMSG
USE SCHEDMSG
USE SECMSG
USE DSKTOG
USE RELTOG
USE PBDREL
USE CHECK
TYPE DISKMSG
USE PBDONLY
USE SAVEPBT
TYPE RSMMSG
USE AUTOUNLD

An <option numeric code> is defined as:

USE OPTN <integer>

where the <integer> used specifies the option.

The S0 message has either of the following formats:

S0 <option mnemonic>

or

S0 <option numeric code>

Examples

S0 TYPE BOJ
S0 USE OPTN 45

The R0 message has either of the following formats:

APPENDIX C (cont)
MESSAGES

RO <option mnemonic>

or

RO <option numeric code>

Examples

RO USE TERMINATE

RO USE OPTN 42

The TO message has the following format:

TO

THE SF MESSAGE

The Multiprocessing Factor can be changed by typing in SF <decimal number> (meaning to set-the-factor). The <decimal number> is defined as in ALGOL, with the restriction that <unsigned integer>s are, at the most, two digits long:

<decimal number> ::= <unsigned integer> | <decimal fraction>
 <unsigned integer> <decimal fraction>

<decimal fraction> ::= .<unsigned integer>

<unsigned integer> ::= <digit> | <digit> <digit>

THE SS ALL MESSAGE

The SS ALL message provides a means whereby a message can be sent to all "remote SPO" users on the system.

The SS ALL message has the following format:

SS ALL: {any characters excluding those having control
 significance}

Example

SS ALL: P.M. STARTS IN 30 MINS.

APPENDIX C (cont)
MESSAGES

THE <mix index> SS ALL MESSAGE

The <mix index> SS ALL message provides a means whereby a message can be sent to "remote SPO" users of a particular program, regardless of whether they have requested messages via the SM message.

If the given mix has no users, the message

NO STATIONS ON MIX = <mix index>

is returned.

The <mix index> SS ALL message has the following format:

<mix index> SS ALL: {any characters aside from those having
control significance}

Example

2 SS ALL: YOU MUST BE OFF BY 0900;
5 MIN. TO GO.

THE SS MESSAGE

The SS message may be used at the central SPO or, if preceded by a question mark, on a remote station with SPO capabilities. It directs a message to a remote station which has SPO capabilities, or to the SPO. If the station addressed is not recognized to have SPO capabilities or is not ready, an INV STN message is returned. The message, as provided at the addressed station, has a prefix which includes the address of the originator.

The SS message has one of the following formats:

SS <remote station address> : <remote station message>
SS SPO : <remote station message>

Examples

SS 1/0 : ARE YOU THERE
? SS SPO : I NEED A SCRATCH TAPE

APPENDIX C (cont)

MESSAGES

THE <mix index> SS MESSAGE

The <mix index> SS message provides a means whereby a message can be sent to all remote SPO users who have requested mix messages via the SM message for a program with a particular <mix index>.

If no users have requested messages, the message

NO SM STATIONS ON MIX = <mix index>

is returned.

The <mix index> SS message has the following format:

<mix index> SS: {any characters aside from those having
control significance}

Example

3 SS: THE TAPE ON MTA IS ALMOST FULL.

THE ST MESSAGE

The ST message allows the system operator to suspend the program referenced by the <mix index> as soon as that program becomes ready to be returned to normal state by the MCP. To resume processing of the program, the operator must use the OK message.

The ST message has the following format:

<mix index> ST

Example

1 ST

THE SV MESSAGE

The SV message may be used to cause a peripheral unit to be made inaccessible until a HALT/LOAD operation occurs or until an RY message referencing the inaccessible unit is entered. If, when the SV message

APPENDIX C (cont)

MESSAGES

is entered and the specified unit is not in use, the message

<unit mnemonic> SAVED

is typed. If a unit is in use when an SV message referencing it is entered, the message

<unit mnemonic> TO BE SAVED

is typed, and the unit becomes inaccessible as soon as it is no longer in use. Until an RY message referencing the unit is entered or a HALT/LOAD occurs, the saved unit remains inaccessible.

The SV message has the following format:

SV <unit mnemonic> <blank> | <unit mnemonic> <unit mnemonic>

Examples

SV LPA

SV MTT

SV CRBMTAMTB

THE SY MESSAGE

The SY message allows the operator to save the Statistics Data Base (MACRO) Log File. (Refer to Statistics Log in section 5.)

The SY message has the following format:

SY

Example

SY

THE TF MESSAGE

The Factor can be interrogated by typing in TF (meaning to type out the Factor).

APPENDIX C (cont)
MESSAGES

THE TI MESSAGE

The TI message causes the MCP to type out the amount of processor time that the subject program has used up at the time the TI message is entered. The time is provided as one to three \langle integer \rangle s separated by \langle space \rangle s, for example:

1
or
2 49
or
1 48 7

The right-most \langle integer \rangle specifies seconds, the second-from-right integer specifies minutes, and the third-from-right integer specifies hours.

The TI message has the following format:

\langle mix index \rangle TI

Example

3 TI

THE TL MESSAGE

The TL message provides a means whereby the MCP types the processor and I/O time limits for a designated job.

The TL message format is:

\langle mix index \rangle TL

THE TR MESSAGE

The TR message allows the system operator to change the value of the time word used by the MCP.

The time, specified by the \langle integer \rangle in the TR message, is designated according to a 24-hour clock, i.e., military time.

APPENDIX C (cont)
MESSAGES

The TR message has the following format:

TR <integer>

Example

TR 0800

THE TS MESSAGE

The TS message makes it possible to determine the programs in the schedule. The MCP types out the names of each job in the schedule, together with the amount of core space needed by the program and the amount of time the program has been in the schedule.

The format of the TS message is:

<priority> <job specifier> = <schedule #> IN FOR
<elapsed time in schedule>, NEEDS <core requirement>

THE UL MESSAGE

The UL message is used in response to a no-file message and allows the system operator to designate the unit on which a particular unlabeled file is located. The unit designated in the UL message may denote the location of a standard file (a file on which the first record is a standard label) or a nonstandard file (a file with no standard label). However, in either case, all records in the file, including the standard label, if any, are recognized as data records. This message differs from the IL message; when the IL message is used in reference to a standard file, a standard label is not recognized as a data record.

The UL message can be used to designate a tape, which does not have a standard library dump tape label, for a library load. However, the tape so designated must conform to the library dump tape format (blocking size) except for the label.

A <mix index> term must be used with the UL message since, during multiprocessing, more than one no-file message may be in effect at the same time.

APPENDIX C (cont)

MESSAGES

The UL message has the following format:

<mix index> UL <unit mnemonic>

Example

1 UL MTT

THE US SPO MESSAGE

The US SPO message inhibits output to the SPO.

The US SPO message has the following format:

US SPO

THE <mix index> WA MESSAGE

The <mix index> WA message provides a means for determining what stations are assigned to a particular program.

If any stations are assigned to the given mix, the MCP returns a message of the form:

<integer>/<integer> ASSIGNED TO <program specifier>

If no stations are assigned to the given mix, the MCP returns the <mix index> WA message preceded by the word NULL.

The <mix index> WA message has the following format:

<mix index> WA

Example

4 WA

THE WD MESSAGE

The WD message causes the MCP to type the date currently being used by the system. The date is given in the MM/DD/YY format.

APPENDIX C (cont)
MESSAGES

The WD message has the following format:

WD

THE WI MESSAGE

The WI message identifies the INTRINSICS currently being used by the MCP.

The following constructs differ in that the first response is typed if no options are set TRUE when the INTRINSICS are compiled.

INTRINSICS (version) . (release level)

INTRINSICS (version) . (release level) INCLUDES
<MCP compiled options>

THE WM MESSAGE

The WM message allows the system operator to request the current version and release-level of the MCP. A list of the module options under which the current MCP has been compiled is also typed.

The WM message has the following format:

WM

Example

WM

Response

MCP XI.14.120 INCLUDES DATACOM, DCSP0, DEBUGGING

THE WP MESSAGE

The WP message provides a means for determining what programs are assigned to what remote stations. If the WP message is followed by tu/buf (where tu and buf are each 1- or 2-digit numbers), the MCP returns a message specifying what programs are assigned to the specified

APPENDIX C (cont)
MESSAGES

stations, if any. If WP alone is entered, the MCP returns a complete list specifying what programs are assigned to what stations.

The message, used to specify what programs are attached to what stations, is as follows:

<integer> / <integer> ASSIGNED TO <program specifier>

If a positive response cannot be provided for a WP message, the message is returned preceded by the word NULL.

The WP message may have either of the following formats:

WP <integer>/<integer>

or

WP

Examples

WP 2/6

WP

THE WR MESSAGE

The WR message is used to create a REMOTE/LOG file on disk. If there is no REMOTE/LOG file on disk when this message is entered, 1000 segments are obtained for the file and it is entered in the Disk Directory.

THE WT MESSAGE

The WT message causes the MCP to type out the time of day currently recognized by the system. The time is given according to a 24-hour clock.

The WT message has the following format:

WT

THE WU MESSAGE

The WU message provides a means for determining the user

APPENDIX C (cont)

MESSAGES

identifications of remote SPO users. If the WU message is preceded by a mix index, the MCP identifies the users of that mix, if any. If the WU is followed by tu/buf (where tu and buf are each 1- or 2-digit numbers), the MCP identifies the user of the given station, if any. If WU is used alone, the MCP identifies all users of remote SPO stations, if any. The message used to identify the user of a remote station is as follows:

`<integer> / <integer> USED BY <user code>`

If no users are referenced by a WU message, the message is returned preceded by the word NULL.

The WU message may have one of the following formats:

```
<mix index> WU
WU <integer>/<integer>
WU
```

Examples

```
3WU
WU 1/4
WU
```

THE WY MESSAGE

The WY message allows the system operator to request that the MCP provide information as to why a program has been temporarily suspended, providing that the program has been temporarily suspended because of a reason previously designated in a system message which:

- a. Is preceded by the character #.
- b. Contains a <job specifier>, e.g., a program which has been suspended because of the condition denoted by a previous # NO FILE message.

APPENDIX C (cont)

MESSAGES

In response to the WY message, the MCP:

- a. Lists the 2-letter codes for all keyboard input messages which can be entered to eliminate the condition that has caused the program to be temporarily suspended.
- b. Retypes the # message that was previously typed to inform the system operator of the condition that caused the program to be suspended.

The WY message has the following format:

<mix index> WY

Example

4WY

THE XS MESSAGE

The XS message causes a program which is in the schedule to be loaded in spite of the fact that the MCP does not think the program will run efficiently with the jobs already in the mix. This is done by typing in the XS message, meaning to execute from the schedule.

The XS message has the following format:

<schedule index> XS

Examples

1XS

2XS

THE XT MESSAGE

The XT message is used to extend the time limits for a job. If <processor part> is <integer>, the processor time limit is extended by <integer> minutes. If either the processor or I/O time limits are extended, a message is printed to notify the operator. In any event, the processor and I/O limits are typed out.

APPENDIX C (cont)

MESSAGES

The XT message format is:

$\langle \text{mix index} \rangle$ XT $\langle \text{processor part} \rangle$ $\langle \text{I/O part} \rangle$
 $\langle \text{processor part} \rangle ::= \langle \text{empty} \rangle \mid \langle \text{integer} \rangle \mid *$
 $\langle \text{I/O part} \rangle ::= \langle \text{empty} \rangle \mid , \langle \text{empty} \rangle \mid , \langle \text{integer} \rangle \mid , *$

APPENDIX D
THE REMOTE SPO STATION FACILITY

GENERAL.

Remote stations are equipped with many of the capabilities of the supervisory printers. Most keyboard input messages can be utilized by the remote user and some system messages, such as NO FILE messages, can be printed on remote stations.

In order for a remote station to make use of SPO facilities, it must demonstrate that it is a typewriter station or TWX. One means whereby the MCP recognizes such a station is due to having received a WRU signal from that station. The WRU signal is automatically generated by a TWX after a remote operator successfully dials the computer. The WRU signal is also generated when the WRU key on a TWX or typewriter is pressed together with the control key. The second way in which a station is recognized as having SPO capabilities is if a log-in (LI) message is entered.

In order for a keyboard input message to be entered from a remote station, the operator is required to add a question mark as a prefix to the message to denote that it is directed to the MCP. Messages without a question mark prefix are assumed to be directed to an object program to which that station is attached.

It should be recognized that certain keyboard input messages are not allowed to be entered from remote typewriters. If such a message is entered, an INV KBD message will be returned.

When control cards are entered from remote stations, the keyboard input message may start with two question marks or a question mark followed by CC; i.e., a control card is entered as at the SPO with the exception that an additional question mark must precede the message.

NOTE

If more than one control card and/or program-parameter card is to be entered and the cards are related to the same program (e.g., an Execute card and label equation cards), the

APPENDIX D (cont)

THE REMOTE SPO STATION FACILITY

cards must all be entered as one message, using the semicolon convention to separate the cards.

Any SPO capable station may be specified to be an alternate SPO providing the station is in ready status at the time the BS message is entered. The station will remain an alternate SPO until a US message is entered for it or until the station goes not ready. A maximum of four alternate SPO's may be active at one time. If a US message has been entered for the SPO and no other SPO consoles are active, the SPO will automatically resume active status.

Input is entered from a typewriter or TWX SPO console as from the SPO. There is no input request, and the left arrow (←) is used as the END-OF-MESSAGE signal. The break key is used as an input request key when the station is typing. The station will then remain idle until input has been entered. Typing will resume with the messages queued for output. The BK message may be used to clear this queue.

Each SPO console gets only those messages it requested or which are necessary for effective system management. The following exception should be noted. If an input request is made to the SPO and the SPO is not active for output, the response to the input request will go to all SPO consoles.

ATTACHING REMOTE STATIONS TO PROGRAMS.

When it is said that a remote station is attached to an object program, it means that some action has been taken to denote that input messages from that station can be read by that object program; i.e., more than one program can be attached to a given station.

If a program wishes to attach a station to itself, it may do so by performing a READ, SEEK or WRITE statement which references that station.

If the operator at a remote station which has SPO capabilities desires to attach his station to a program, he may do so by entering an EXECUTE or RUN card for that program.

APPENDIX D (cont)
THE REMOTE SPO STATION FACILITY

A \langle remote station address \rangle is defined as:

\langle integer \rangle / \langle integer \rangle

where the first \langle integer \rangle specifies the number of the terminal related to the remote station being referenced, and the second \langle integer \rangle specifies the relevant buffer number.

A \langle remote station message \rangle is defined as:

a string of characters, the end of which is recognized to be a group mark, i.e., a left arrow (\leftarrow) or END OF MESSAGE.

THE RUN CARD.

The RUN card is provided for use at remote data communications stations which have SPO capabilities. The purpose of the RUN card is to provide the operator of such a data communications station with the ability to attach his station to a program.

If, when a RUN card is recognized by the MCP, the designated program is in the mix, the attaching process alone takes place and the message

\langle job specifier \rangle RUNNING

is given in response to the RUN card. However, if the specified job is not in the mix, the program is scheduled and executed in the normal fashion. The station is then attached to the program.

The RUN card must contain the following information:

? RUN \langle program specifier \rangle \langle comment \rangle

Example:

? RUN MANYSTA/HANDLER

ADDITIONAL KEYBOARD INPUT MESSAGES FOR REMOTE STATIONS.

The following paragraphs describe messages made available for use at remote stations which have SPO capabilities.

APPENDIX D (cont)
THE REMOTE SPO STATION FACILITY

THE BS MESSAGE.

The BS message sets the station indicated by <remote specifier> as a SPO console.

The BS message has the following format:

BS <remote specifier>

THE HR MESSAGE.

The HR message is used to detach a station from a program. It can be considered to be the opposite of a RUN card.

The HR message has the following format:

<mix index> HR

Example:

1 HR

THE LI MESSAGE.

The primary purpose of the LI message is to require that a remote operator identify himself as a legitimate user of the system in order for the MCP to allow him to make use of the system.

If an LI message is entered while a remote operator is already logged in, the MCP logs out the previous user before logging in the new user.

The LI message has the following format:

LI <separator> <user code> <separator> <authentication code>

Examples:

LI : 007 : M
LI BY CHARLEY

THE LO MESSAGE.

The LO message is provided so that a remote user may log out after having logged in. This is desirable in that anyone who attempts to

APPENDIX D (cont)

THE REMOTE SPO STATION FACILITY

use the remote typewriter, subsequent to the departure of the proven legitimate user, must also be a legitimate user.

APPENDIX D (cont)
THE REMOTE SPO STATION FACILITY

The LO message must have the following format:

LO

THE SM AND HM MESSAGES.

The <mix> SM message requests MIX messages for a given job. MIX messages for other jobs attached to a station are not affected.

The SM message requests MIX messages for all jobs which originate from a control console. This request is assumed at log-in time and remains in effect until an HM message is entered.

The <mix> HM message turns off the request for MIX messages for a given job. MIX messages for other jobs attached to a station are not affected.

The HM message requests that no MIX messages be given for jobs originating from a control station. This request remains in effect until an SM message is entered or until the station is logged out and logged in again.

The SM message may have either of the following formats:

SM

or

<mix index> SM

The HM message may have either of the following formats:

HM

or

<mix index> HM

Examples:

SM

1 SM

APPENDIX D (cont)
THE REMOTE SPO STATION FACILITY

HM
1 HM

THE SS MESSAGE.

The SS message may be used at the central SPO or, if preceded by a question mark, on a remote station with SPO capabilities to direct a message to a remote station which has SPO capabilities, or to the SPO. If the station addressed is not recognized to have SPO capabilities or is Not Ready, an INV STN message is returned. The message, as provided at the addressed station, has a prefix which includes the address of the originator.

The SS message has the following format:

SS <remote station address> : <remote station message>

or

SS SPO : <remote station message>

Examples:

SS 1/0 : ARE YOU THERE
?SS SPO : I NEED A SCRATCH TAPE

THE US MESSAGE.

The US message un-sets the station indicated by <remote specifier> from SPO console status.

The US message has the following format:

US <remote specifier>

ALTERNATE SPO CONSOLES:

Any SPO capable station may be specified to be an Alternate SPO providing that it is ready at the time the BS message is entered. It will remain an alternate SPO until a US message is entered for it or until the station goes not ready. Up to four alternate SPO's may be active at one time. There is provision made so that the SPO will automatically resume active status if there are no other active SPO

APPENDIX D (cont)
THE REMOTE SPO STATION FACILITY

consoles, even though an US message may be entered for it.

Input is entered from a typewriter or TWX SPO console as from the SPO; however, there is no input request button, and the left arrow (←) is used as the end-of-message signal. The break key is used as an input request key when the station is typing. When this is used, the station will remain idle until there has been input entered. It then will resume with the messages queued for output. (The BK message may be entered to clear this queue.)

Each SPO console gets only those messages which it requested or which are necessary for effective system management. The exception to this is that if the SPO is not active for output, any Keyboard request response to input from the SPO will go to all SPO consoles.

APPENDIX E
FILE SECURITY SYSTEM

ADDITIONS TO THE DATA COMMUNICATIONS MCP.

LOG-IN PROCEDURE FOR DATA COMMUNICATIONS.

Before a user can utilize the MARK III Data Communications system capabilities, he will first be required to "log-in" to the system by entering a Log-In (LI) message. This message will contain a "remote user identification" consisting of an identification code and, at the option of the installation, an authentication code which is not <empty>. It is necessary to note that more than one "remote user identification" can contain the same user authentication code. In general, however, each "remote user code" must be unique.

The MCP will check the "remote user identification" contained in the Log-In message against the file of authorized user codes called REMOTE/USERS. If the code entered is not in the file, the "user" will be so informed by the MCP and will be prevented from using the system until a correct Log-In message is typed in. If the code entered is in the list of authorized users, the MCP will log the station in, record its log-in time, and consider the station to be a bona fide user. Also, this user's station will be considered to have "SPO capabilities."

A remote station having SPO capabilities can enter system keyboard input messages, including control card information. If a remote user is properly logged-in, three masks will be assigned to his station. These masks can be used to limit both the control card and keyboard input messages that this user is allowed to enter at his station, if the masks have been added to the REMOTE/USERS. If no list of authorized users has been supplied by the particular installation, then a user will be logged-in regardless of the code entered, but no log-in time will be recorded. In this case, the standard masks defined in the MCP will be used.

The Log-In message has the following form:

? LI <separator> <user code> <separator>
 <authentication code>

APPENDIX E (cont)
FILE SECURITY SYSTEM

Examples:

? LI : 007 : M
? LI BY CHARLEY

USER CONTROL CARD.

A USER Control Card is used to enter a user code from an input source other than a remote station. USER cards received from a remote station are ignored. The user code from a USER card is used to initialize a job's entry in an MCP table of user codes when a job is selected to run. A USER card must precede an EXECUTE card, and, if there is more than one USER card per deck, only the first card will have any significance. If no USER card is entered, a job's user table entry will be initialized to zero.

The following information must appear on a USER control card:

?USER=<user code>

Examples:

?USER=BATMAN
?USER=SUPERMAN

USER CODE AND AUTHENTICATION CODE.

The user code is used to verify that a job in the system has been authorized to use a particular disk file. Checking for an authorized user is done by the MCP at file open or close time. A user code may be presented to the system via a USER Control Card at the central site card reader, or via the log-in procedure for remote stations. The MCP checks to make sure that only authorized users are permitted to use the system.

The basic idea of the file security system is that the creator of a file must specify which users are authorized to use his file. This implies that a person must divulge his user code to anyone who has created files he wants to use. Therefore, one's user code cannot be

APPENDIX E (cont)
FILE SECURITY SYSTEM

kept completely confidential. For this reason, a person at a remote station may be required to type-in a log-in message which contains his "remote-user identification", which consists of his particular authentication code as well as his user code. The authentication code would then be used by the MCP to verify that the person attempting to get access to the system is in fact authorized to use the specified user code. This authentication code need never be divulged to anyone else. Although an authentication code may be <empty>, it can be used to provide a significant increase in the security of the system.

User and authentication codes are restricted in size to seven alphanumeric characters. Any code longer than seven characters will be truncated to seven characters.

<user code> ::= <letter> | <digit> | <user code> <letter> |
 <user code> <digit>

<authentication code> ::= <letter> | <digit> |
 <authentication code> <letter> |
 <authentication code> <digit> |
 <empty>

<letter> ::= {any of the 26 characters of the alphabet}
<digit> ::= 0|1|2|3|4|5|6|7|8|9

Examples:

BOND
CHARLEY
GOLDFINGER (truncated to GOLDFIN)
4UIWILL

LEVELS OF SECURITY.

The file security system has been designed to prohibit unauthorized users from having access to the system or to any files belonging to authorized users. The system is based on the idea that files may be

APPENDIX E (cont)
FILE SECURITY SYSTEM

made "private" by a "security file" which contains a list of authorized users and programs which may access the file. Only the creator of a file may establish and maintain the security file associated with his file. A user trying to use a "private" file will be checked by the MCP against the list before the requestor is allowed to access any records contained in the file. If neither the user's user code nor the program specifier of the program being executed are contained in the security file for the file being opened, then program control will be transferred to either the parity action label (for ALGOL programs) or to the USE routine (for COBOL programs). The absence of an error handling label or routine will result in the termination of the program.

Once the requestor is defined as being privileged (unlimited access), primary (unlimited access to files created by this user), secondary (access to the file for input or output), or tertiary user (access for input only), the file will be made available to the requestor, and the MCP will note the manner in which this requestor may access the file. Access to the file in any unauthorized manner will result in either transfer of control to the error handling label or routine, or termination of the program.

IDENTIFIED USER.

A user may be identified by either logging-in from a remote station or entering a USER control card. An identified user may open a private file for input if he is a privileged, primary, secondary or tertiary user. An identified user may open a private file for input/output if he is a privileged, primary or secondary user. An identified user may maintain a private file (i.e., perform library or security file maintenance) if he is either the privileged or primary user. A user is considered the privileged user if his user code is the first entry in the REMOTE/USERS file.

$\langle \text{privileged user} \rangle ::= \{ \text{a user who has unlimited access.} \}$
 $\langle \text{primary user} \rangle ::= \{ \text{the creator of the file; i.e., the user who caused the file to be entered into the disk directory.} \}$

APPENDIX E (cont)
FILE SECURITY SYSTEM

- <secondary user> ::= {a user designated as being able to access a file for input or output}
- <tertiary user> ::= {a user designated as being able to access a file for input only}
- <security file> ::= {the file containing user codes for the secondary and/or tertiary users}
- <free file> ::= {a file open to all users for input, output, and library maintenance, and to the privileged user only for security maintenance}
- <public file> ::= {a file open to all users except for library or security file maintenance}
- <private file> ::= {a file with an associated security file}
- <sole-user file> ::= {a file whose only valid user is its primary user}
- <unlocked file> ::= {a file which may be read by all users; however, only the privileged or primary user may write on it}

FILE HEADERS.

The Disk Directory file header of a nonfree file contains the user code of the primary user (creator) of the file and the name of the associated security file, if any.

APPENDIX E (cont)
FILE SECURITY SYSTEM

Format of File Header for Various Types of Files

<u>Type</u>	<u>Word 2</u>	<u>Word 5</u>	<u>Word 6</u>
Free File	0	[0:42] = 0 [42:6] = "?" (OCT14)	[0:42] = 0 [42:6] = "?" (OCT14)
Sole User File	[1:1] = 0 [6:42] = primary user's user code	0	0
Public File	[1:1] = 0 [6:42] = primary user's user code	[0:42] = 0 [42:6] = "?" (OCT14)	0
Private File	[1:1] = 0 [6:42] = primary user's user code	[1:1] = 1 [6:42] = multi- file id of se- curity file	[1:1] = 0 [6:42] = file id of secur- ity file
Security File	[1:1] = 1 [6:42] = primary user's user code	0	0
Unlocked File	[1:1] = 0 [6:42] = primary user's user code	[0:42] = 0 [42:6] = "?" (OCT14)	[0:42] = 0 [42:6] = "?" (OCT14)

APPENDIX E (cont)
FILE SECURITY SYSTEM

FORMAT OF SECURITY FILE ENTRIES.

Entries in a security file must be one word in length to specify a user code, deleted entry, or last entry. An entry for a program specifier¹ must be two contiguous words in length and both words must be contained in the same record. Hence, the program identifier cannot fall in the last word of a record.

Each one-word entry and the first word of a two-word entry must conform to one of the following formats.

<u>Bits [1:5]</u>	<u>Bits [6:42]</u>
00000	User code of a secondary user
10000	User code of a tertiary user
00011	Program identifier of a program which may access the file for input or output; i.e., in the same manner as a secondary user
00010	Program identifier of a program which may access the file for input only; i.e., in the same manner as a tertiary user
00000	12 (decimal) to indicate a deleted entry
00000	76 (decimal) to indicate the last entry

The second word of a two-word entry must conform to the following format:

<u>Bits [1:5]</u>	<u>Bits [6:42]</u>
00000	Program identifier suffix

¹ A program specifier is defined as:
 ⟨program specifier⟩ ::= ⟨program identifier⟩ ⟨separator⟩
 ⟨program identifier suffix⟩

APPENDIX E (cont)
FILE SECURITY SYSTEM

MCP ACTIONS.

The MCP maintains a table of user codes, called the USERCODE table, for each active mix index. This table contains the contents of the user code from either the log-in (LI) message or the USER control card. The table is used for the purpose of file protection whenever a disk file is opened or closed with a lock or purge.

In addition to the USERCODE table, the MCP maintains a table containing user codes associated with each active (logged in) terminal buffer. The Data Communications Interrogate function has been extended to update the USERCODE table with the user code associated with the specified terminal buffer. This facility is necessary to allow a program which may handle more than one user to create and/or access files for any of those users.

MCP actions for files (open input and output) already in the Disk Directory are as follows:

- a. All unlocked, free, and public files are made available to any user.
- b. Sole-user files are available only to the primary user of the file.
- c. Accessing any private file requires that an entry in the associated security file be equivalent to the entry of the requesting job in the USERCODE table or to the program specifier of the requesting program. If verification cannot be made, transfer is made to the parity action label (ALGOL) or the USE routine (COBOL), if one is present. Otherwise, the program is terminated.

MCP actions for files (close output, close lock) not in the Disk Directory are as follows:

- a. If the USERCODE table entry is empty, the file is made a free file.

APPENDIX E (cont)
FILE SECURITY SYSTEM

- b. Otherwise, the file is entered into the Disk Directory as a sole-user file.

For ZIP (<program specifier>), the MCP appends the user code from the USERCODE table to the control card information. For the use of the generalized ZIP, the MCP inserts the user code from the USERCODE table into the array or the file header.

Any successive user codes entered via the USER control card are ignored.

LIBRARY AND SECURITY FILE MAINTENANCE.

Library maintenance functions require the presence of a user code for maintenance of any nonfree file. The user code must be introduced via a USER control card or log-in message which must precede the library maintenance control cards. Library maintenance is not performed on nonfree files unless the user code is equivalent to either the privileged user or the primary user (creator) of the file.

For the DUMP function, the MCP automatically dumps the associated security file as well as the specified private file.

■ Five Security File maintenance functions are provided to maintain security files and private files. These functions require the presence of a user code entered via a USER control card or log-in message preceding the Security File maintenance information.

- a. USE Control Card.

?USE <security file specifier> ON <file specifier>

The <file specifier> is made into a private file using the <security file specifier> as its security file. The <security file specifier> names a file which must have the security file format and must be a sole-user file. This file may or may not be a security file for other files at this time. The <file specifier> must be a sole-user file, and the user code of the requester must be equivalent to the primary user (creator) of the file.

APPENDIX E (cont)
FILE SECURITY SYSTEM

b. LOCK Control Card.

?LOCK <file specifier>

The <file specifier> is made into a sole-user file. The <file specifier> must be either a private, unlocked, or public file; and the user code of the requester must be equivalent to either the privileged or primary user of the file. In addition, the privileged user may LOCK a FREE file.

c. PUBLIC Control Card.

?PUBLIC <file specifier>

The <file specifier> is made into a public file. The <file specifier> must be a private, unlocked, or sole-user file; and the user code of the requester must be equivalent to either the privileged or primary user of the file.

d. FREE Control Card.

?FREE <file specifier>

The <file specifier> is made into a free file. The <file specifier> must be either a sole-user, private, unlocked, or public file; and the user code of the requester must be equivalent to either the privileged or primary user of the file.

e. UNLOCK Control Card.

?UNLOCK <file specifier>

The <file specifier> is made into an unlocked file. The <file specifier> must be a public, private, or sole-user file; and the user code of the requester must be equivalent to either the privileged or primary user of the file.

Examples:

?USER=BOSS; USE SECURE/BLOCK ON ALGOL/=,DIRCTRY/DISK;END.

?USER=BOSS; LOCK COBOL/DISK,LOGOUT/DISK;END.

?USER=CHARLEY; PUBLIC CHARLEYS/FILES;END.

?USER=1234567; FREE PUBLIC/FILES;END.

APPENDIX E (cont)
FILE SECURITY SYSTEM

SECMSG AND DSKTOG OPTIONS.

The typing of Security File maintenance messages is controlled by OPTN 29, SECMSG. These messages can be enabled by setting this option, i.e.,

SO USE SECMSG.

Security File maintenance messages can be inhibited by resetting this option, i.e.,

RO USE SECMSG.

The setting of OPTN 29 is RESET. It is necessary, therefore, to SET this option in order to get the Security File maintenance messages.

■ OPTN 28, DSKTOG, if SET, prevents an object program from performing any I/O operation at any absolute disk address less than the starting address of USER DISK. If an attempt is made to perform a disk operation on any address below the beginning of USER DISK, the message:

-INVALID PRL <program specifier> , <terminal reference>

is typed out and the program is terminated. This is sufficient to keep an object program from referencing the Disk Directory.

If OPTN 28 is RESET, a check is not made, and the MCP behaves exactly as it has heretofore.

The setting of the DSKTOG option, OPTN 28, is RESET. It is necessary, therefore, to SET the option in order to have the Directory protection check in force.

APPENDIX E (cont)
FILE SECURITY SYSTEM

It should be restated that if the option is RESET, the entire File Security system can be bypassed in a relatively straightforward manner. If the option is SET, programs which now reference the Disk Directory as a file will not run and will have to be revised to use the Directory Search Statement.

REMOTE/USERS FILE.

A file called REMOTE/USERS will be maintained on the disk and will contain the list of authorized users. The only valid users of this file are the designated "privileged user" and the MCP. This file can be created and maintained by the privileged user with an ALGOL program, an example of which, adequate for most installations, is provided by Burroughs. This program is called UPDATE/USERS and is described in subsequent paragraphs.

A record in the REMOTE/USERS file will contain at least the following information:

- a. User identification code.
- b. Control card mask.
- c. Information keyboard input message mask.
- d. Mix-related keyboard input message mask.

At the option of each installation, an authentication code and/or comments can also be included in a record. The record size must be constant throughout the file and must be either 6, 10, 15, or 30 words. Records that are deleted from the file must have a user code value of 76.

The format of a REMOTE/USERS record is:

- a. Valid entry:

<u>Word</u>	<u>Contents</u>
0	User identification code
1	Control card mask no. 1
2	Control card mask no. 2
3	Information message mask no. 1

APPENDIX E (cont)
FILE SECURITY SYSTEM

<u>Word</u>	<u>Contents</u>
4	Information message mask no. 2
5	Mix-related message mask
(The following entries are optional)	
6	If < 0, then authentication code If > 0, then available to user
7-to-recordsize	Available to user

b. Deleted entry:

<u>Word</u>	<u>Contents</u>
0	12 (octal value 014)
1-to-recordsize	Irrelevant

c. Last record in file:

<u>Word</u>	<u>Contents</u>
0	76 (octal value 114)
1-to-recordsize	Irrelevant

UPDATE/USERS PROGRAM.

The UPDATE/USERS program is an ALGOL program which can be used to create and maintain the REMOTE/USERS file. It is provided so that installations will have an immediate capability to create and maintain a REMOTE/USERS file. Although this program will be sufficient for most installations, it is also intended to serve as a guide for those installations wishing to write an installation-oriented program to maintain the REMOTE/USERS file in a specialized manner.

The UPDATE/USERS program will create a new REMOTE/USERS file if none exists at the time it is executed. Alternatively, it will update the REMOTE/USERS file if one does exist at the time it is executed.

If a REMOTE/USERS file does exist, the record size of the file will not be changed by the update run. If a new REMOTE/USERS file is being created, it is possible to specify the size of the records

APPENDIX E (cont)
FILE SECURITY SYSTEM

in the file by providing a COMMON value to the program. This COMMON value should be either 6, 10, 15, or 30. If a COMMON value is not provided or if the COMMON value is not one of the four integers mentioned, the file is created with 30-word records.

The data deck for the UPDATE/USERS Program specifies the user codes for which some action is to be taken.

An ADD card is used, during either a creation or update run, to insert a new "remote user identification" into the file. Following the ADD card are mask cards which specify which control cards and keyboard input messages are allowable for this user. These cards are the CCMASK card which specifies the control cards that are allowable, the INFOMASK card which specifies the information keyboard input messages (e.g., MX, WT) that are allowable, and the MIXMAX card which specifies the mix-related keyboard input messages (e.g., nTI, nDS) that are allowable. Associated with each of the mask card types is a standard mask which is assumed if a card of that type is not present. Thus, if no CCMASK card follows an ADD card, the standard control card mask is assumed. Also, if more than one mask card of a given type appears after an ADD card, only the last card of that type is used.

An ADD card can also be used during an update run to change the mask information associated with the user. If the "remote user identification" on an ADD card matches a "remote user identification" in the REMOTE/USERS file, the mask cards following the ADD card are used to specify the mask information for that user, overwriting the previous mask information.

A DELETE card is used to remove a user from the REMOTE/USERS file. This is done only if the "remote user identification" on a DELETE card matches a "remote user identification" in the REMOTE/USERS file. Mask cards are not used following a DELETE card.

The ADD and DELETE functions are handled on a one-at-a-time basis. Therefore, it is possible to first delete a "remote user identification" and then put it back into the file (perhaps with a different authentication code) in the same execution of the UPDATE/USERS Program.

APPENDIX E (cont)
FILE SECURITY SYSTEM

Also, if several ADD cards contain the same "remote user identification," only the mask cards behind the last ADD card are effective.

The ADD card format is:

ADD <user code> <separator> <authentication code>

The above information may appear anywhere within columns 1-72.

Columns 73-80 must be blank. The <authentication code> may be <empty> and is valid only if the record size is greater than 6.

The DELETE card format is:

DELETE <user code> <separator> <authentication code>

The above information may appear anywhere within columns 1-72.

Columns 73-80 must be blank. An <authentication code> which is not <empty> is only required if the record to be deleted contains an <authentication code>.

The CCMASK card format is:

A Yes in a column indicates that the corresponding control card is to be allowed; a No in a column indicates that the corresponding control card is not to be allowed.

<u>Column</u>	<u>Word</u>	<u>CCMASK1 Bit</u>	<u>Standard Mask</u>
1-23	Not Used	0-22	--
24	UNLOCK	23	no
25	USE	24	no
26	LOCK	25	no
27	FREE	26	no
28	PUBLIC	27	no
29	USER	28	no
30	RUN	29	yes
31	COMPILE	30	yes
32	EXECUTE	31	yes
33	DUMP	32	no
34	UNLOAD	33	no
35	ADD	34	no
36	LOAD	35	no
37	REMOVE	36	no
38	CHANGE	37	no
39	UNIT	38	no

*THIS IS
ALL OBSOLETE
SEE SYSTEM NOTE 14*

APPENDIX E (cont)
FILE SECURITY SYSTEM

<u>Column</u>	<u>Word</u>	<u>CCMASK1 Bit</u>	<u>Standard Mask</u>
40	END	39	yes
41	DATA	40	no
42	LABEL	41	no
43	SET	42	no
44	RESET	43	no
45	Not Used	44	--
46	Not Used	45	--
47	Not Used	46	--
48	Not Used	47	--

<u>Column</u>	<u>Word</u>	<u>CCMASK2 Bit</u>	<u>Standard Mask</u>
49	EXPIRED	0	no
50	ACCESS	1	no
51	PROCESS	2	yes
52	IO	3	yes
53	PRIORITY	4	no
54	COMMON	5	yes
55	CORE	6	no
56	STACK	7	no
57	SAVE	8	yes
58	Not Used	9	--
59	Not Used	10	--
60	Not Used	11	--
61	ALGOL	12	yes
62	XALGOL	13	no
63	FORTRAN	14	yes
64	LONGALG	15	yes
65	BASIC	16	no
66	Not Used	17	--
67	WITH	18	no
68	COBOL	19	yes
69	LIBRARY	20	no
70	SYNTAX	21	no
71	FROM	22	no
72	TO	23	no

OBSOLETE
SEE SYSTEM NOTE 15

In addition, CCMASK must appear within columns 73-80. If a CCMASK card is ^{NOT} entered for an ADD, the standard mask is used.

The INFOMASK card format is:

A Yes in a column indicates that the corresponding message is to be allowed; a No in a column indicates that the corresponding message is not to be allowed.

APPENDIX E (cont)
FILE SECURITY SYSTEM

<u>Column</u>	<u>Word</u>	<u>INFOMASK1 Bit</u>	<u>Standard Mask</u>
1	Not Used	0	--
2	PG	1	no
3	MX	2	yes
4	DD	3	no
5	RW	4	no
6	PD	5	yes
7	DB	6	no
8	DP	7	no
9	DT	8	no
10	DS	9	no
11	PT	10	no
12	RS	11	no
13	EI	12	yes
14	CC	13	yes
15	PB	14	no
16	RY	15	no
17	TR	16	no
18	OL	17	yes
19	LN	18	no
20	WD	19	yes
21	WT	20	yes
22	LR	21	no
23	RO	22	no
24	SO	23	no
25	TO	24	yes
26	SV	25	no
27	LD	26	no
28	CD	27	yes
29	RD	28	no
30	RN	29	no
31	ED	30	no
32	CI	31	no
33	TE	32	yes
34	SF	33	no
35	TS	34	yes
36	RR	35	no
37	QV	36	no
38	EX	37	yes
39	PI	38	yes
40	LO	39	yes
41	LI	40	yes
42	SS	41	yes
43	SM	42	yes
44	HM	43	yes
45	TC	44	yes
46	ZZ	45	yes
47	BO	46	yes
48	WP	47	no

*OBSOLETE
SEE SYSAGTE 14*

<u>Column</u>	<u>Word</u>	<u>INFOMASK2 Bit</u>	<u>Standard Mask</u>
49	Not Used	0	--
50	WU	1	no

APPENDIX E (cont)
FILE SECURITY SYSTEM

<u>Column</u>	<u>Word</u>	<u>INFOMASK2 Bit</u>	<u>Standard Mask</u>
51	LF	2	no
52	LC	3	no
53	LS	4	no
54	XI	5	no
55	WR	6	no
56	WM	7	yes
57	BK	8	no
58	BS	9	no
59	US	10	no
60	SC	11	no
61	CL	12	no
62	QT	13	no
63	WI	14	no
64	CU	15	no

In addition, INFOMASK must appear within columns 73-80. If no INFO-MASK card is entered for an ADD, then the standard mask will be used.

*OBSOLETE
SEE SYNTAX 14*

The MIXMASK card format is:

A Yes in a column indicates that the corresponding message is to be allowed; a No in a column indicates that the corresponding message is not to be allowed.

<u>Column</u>	<u>Word</u>	<u>MIXMASK Bit</u>	<u>Standard Mask</u>
1	Not Used	0	--
2	DS	1	no
3	IL	2	no
4	OU	3	no
5	OK	4	no
6	FM	5	no
7	AX	6	no
8	FR	7	no
9	OF	8	no
10	TI	9	yes
11	WY	10	yes
12	RM	11	no
13	UL	12	no
14	ST	13	no
15	IN	14	no
16	OT	15	yes
17	QT	16	no
18	PR	17	no
19	PS	18	no
20	XS	19	no
21	ES	20	no
22	SM	21	yes
23	HR	22	yes

APPENDIX E (cont)
FILE SECURITY SYSTEM

<u>Column</u>	<u>Word</u>	<u>MIXMASK Bit</u>	<u>Standard Mask</u>
24	CT	23	no
25	XT	24	no
26	TL	25	no
27	SS	26	no
28	WU	27	no
29	WA	28	no
30	HM	29	no
31	CU	30	no

In addition, MIXMASK must appear within columns 73-80.

If no MIXMASK card is entered for an ADD then the standard mask will be used.

The END card format is:

An END card must be used to denote the end of the data deck.

END can appear anywhere within columns 1-72.

Example of an UPDATE/USERS data deck.

? EXECUTE UPDATE/USERS; COMMON=15.

? DATA CODES.

ADD BOSS

111111111111111

11111

11

CCMASK

111111111111111111111111111111111111 111111111 111 111111111111 11111111111

INFOMASK

1111111111111111111111111111

MIXMASK

THESE EIGHT WORDS ARE AVAILABLE FOR COMMENTS.....

ADD SIMPLE THE SIMON

11111 11

1 1 1

1

CCMASK

1 11

111 111 1111

1

INFOMASK

1 1 11

MIXMASK

ADD ALL THE OTHERS

THESE USERS WILL BE ENTERED WITH THE STANDARD MASKS

ADD PRIMO

THIS USER ONLY NEED LOG-IN WITH HIS FIRST NAME

END

? END.

? EXECUTE UPDATE/USERS

? DATA CODES.

APPENDIX E (cont)
FILE SECURITY SYSTEM

DELETE PRIMO
ADD CHARLEY/BROWN
ADD LUCY
END
?END.

ADDITIONAL KEYBOARD INPUT AND SYSTEM MESSAGES.

Incorporation of the File Security System has necessitated the addition of new Keyboard Input and System Messages. These are described below.

THE TC MESSAGE.

The TC message causes the Data Communications MCP to type out the amount of elapsed time that the station was logged-in with an identified user. The time is provided as one to three <integer>'s separated by colons.

APPENDIX E (cont)
FILE SECURITY SYSTEM

Examples:

5

or

7:37

or

2:48:59

The right-most \langle integer \rangle specifies seconds, the second from the right \langle integer \rangle specifies minutes, and the third from the right \langle integer \rangle specifies hours.

The TC message has the following format:

?TC

The MCP responds by typing out a message of the form:

TIME SINCE LOG-IN IS XX:XX:XX

CHANGES TO THE PD MESSAGE.

The PD (Print Directory) message has been changed to type out only the names of free or public files, or those files for which the requestor is either the primary, secondary, or tertiary user. In essence, this change means that a person at a remote site will not be able to determine the name of, or even the existence of, any file he may not reference.

This change has no effect if the PD message is entered at the SPO at the central site since the user code for the SPO is assumed to be the user code of the privileged user.

THE LF, LC, AND LS MESSAGES.

The LF, LC and LS messages allow the system operator to request that the Data Communications MCP type information pertaining to the

APPENDIX E (cont)
FILE SECURITY SYSTEM

creation or security of files listed in the Disk Directory. The formats of the messages are shown below. The action caused by the messages depend upon the format of the message.

LF (LIST FILES). The action caused by the LF message is as follows:

- a. If the message of the form

LF =/=

is entered, all <file specifier>'s of non-free files and their primary users (creators) in the Disk Directory will be listed.

- b. If the message of the form

LF <primary user specifier>

is entered, the <file specifier> and <primary user specifier> will be typed out for all files for which the <primary user> is the primary user. If no files can be found for which the <primary user specifier> is the creator, the message

NULL LF <primary user specifier>

will be typed out.

LC - (LIST CREATOR). The actions caused by the LC message are as follows:

- a. If the message of the form

LC =/=

is entered, <file specifier>'s of all non-free files and their creators in the Disk Directory will be listed.

APPENDIX E (cont)
FILE SECURITY SYSTEM

- b. If the message of the form

LC <file specifier>

is entered and the <file specifier> is in the Disk Directory, and non-free, the <file specifier> and its creator will be listed.

- c. If the <file specifier> in the message is not in the Disk Directory, or if the <file specifier> is a free file, the message

NULL LC <file specifier>

will be typed out.

- d. If the message of the form

LC =/ <file identification>

or

LC =/ <program identifier suffix>

is entered, the names of all non-free files in the Disk Directory which have the designated <file identification> or <program identifier suffix> and their creators will be typed out.

- e. If no such files exist in the Disk Directory, a message of the form

NULL LC =/ <file identification>

or

NULL LC =/ <program identifier suffix>

will be typed out.

APPENDIX E (cont)
FILE SECURITY SYSTEM

f. If a message of the form

LC <file identification prefix>/=

or

LC <file identification prefix>

or

LC <program identifier>/=

or

LC <program identifier>

is entered, the names of all non-free files in the Disk Directory which have the designated <file identification prefix> or <program identifier> and their creators will be typed out.

If no such files exist in the Disk Directory, a message of the form

NULL LC <file identification prefix>/=

or

NULL LC <file identification prefix>

or

NULL LC <program identifier>/=

or

NULL LC <program identifier>

will be typed out.

APPENDIX E (cont)
FILE SECURITY SYSTEM

LS (LIST SECURITY FILE). The actions caused by the LS message are as follows:

- a. If the message of the form

LS =/= :

is entered, all private <file specifier>'s and their <security file specifier>'s in the Disk Directory will be typed out.

- b. If the message of the form

LS <file specifier>

is entered and the <file specifier> is in the Disk Directory and is a private file, the <file specifier> and its <security file specifier> will be typed out. If the <file specifier> is not in the Disk Directory or the <file specifier> is not a private file, a message of the form

NULL LS <file specifier>

will be typed out.

- c. If the message of the form

LS =/ <file identification>

or

LS =/ <program identifier suffix>

is entered, the names of all private files in the Disk Directory which have the designated <file identification> or <program identifier suffix> and their <security file specifier> will be typed out. If no such files exist in the Disk Directory, a message of the form

APPENDIX E (cont)
FILE SECURITY SYSTEM

NULL LS =/ <file identification>

or

NULL LS =/ <program identification suffix>

will be typed out.

- d. If a message of the form

LS <file identification prefix>/=

or

LS <file identification prefix>

or

LS <program identifier>/=

or

LS <program identifier>

is entered, the names of all private files in the Disk Directory which have the designated <file identification prefix> or <program identifier> and their <security file specifier>'s will be typed out.

If no such files exist in the Disk Directory, a message of the form

NULL LS <file identification prefix>

or

NULL LS <program identifier>/=

or

NULL LS <program identifier>

will be typed out.

APPENDIX E (cont)
FILE SECURITY SYSTEM

To summarize, the LF message may have any one of the following formats:

LF =/=
LF <primary user specifier>

The LC and LS message may have any one of the following formats:

XX =/=
XX <file specifier>
XX =/ <file identification>
XX =/ <program identifier suffix>
XX <file identification prefix> /=
XX <program identifier> /=
XX <program identifier>

where LC or LS may be substituted for XX's.

THE BO (BLACK OUT) MESSAGE.

In those instances where it is necessary to maintain security in regard to the user and authentication codes, the MCP will (upon request) black out a sufficient number of characters so that the log-in message can be typed over the blacked-out spaces. This feature can be used to make it extremely difficult for the log-in codes to be read.

The black-out feature is invoked by typing in B0 before logging-in. This message will cause the MCP to black out a line on the remote typewriter, leaving the carriage positioned at the beginning of the blacked-out area.

ADDITIONAL SYSTEM MESSAGES.

#STATION<terminal unit>/<buffer address>:NULL VERIFICATION

This message indicates that a user attempted to log-in at a remote station and no verification could be made with the log-in codes and the list of allowable users.

APPENDIX E (cont)
FILE SECURITY SYSTEM

-INVALID PRL<job specifier>,<terminal reference>

This message indicates that an object program has either attempted an input or output operation on a disk file using release logic, or has attempted an input or output operation on a disk address less than the starting address of USER DISK when the DSKTOG option was SET. Consequently, processing of the program was discontinued.

-INVALID USER<file designator>:<job specifier>,<terminal reference>

This message indicates that an object program has attempted an input or output operation on a disk file for which it was not a valid user, and the object program did not specify any action for such a condition. Consequently, processing of the program was discontinued.

<file designator> SECURITY MAINT IGNORED

This message indicates that an attempt was made to perform Security File Maintenance on a disk file that was in a state such that the required maintenance could not be completed.

<user code> INVALID USER OF <program specifier>

This message indicates that an attempt was made to access a program file by a designated user who was not a valid user for the file.

<user code> INVALID USER OF <data file identifier>

This message occurs when an attempt is made to RM on a disk file in response to a duplicate file message. The program which is attempting to close the duplicate disk file is running under a user code which is not a primary user of the existing disk file. This message will not occur if the RM is done at the SPO.

<file specifier> SECURED WITH <file specifier>

This message is typed after the MCP has performed an operation specified by a USE Control Card, providing that the SECMSG option (OPTN 29) is SET.

APPENDIX E (cont)
FILE SECURITY SYSTEM

<file specifier> LOCKED FROM <file specifier>

This message is typed after the MCP performs an operation specified by a LOCK control card, providing that the SECMSG option (OPTN 29) is SET.

<file specifier> FREE FILE

This message is typed after the MCP performs an operation specified by a FREE control card, providing that the SECMSG option (OPTN 29) is SET.

<file specifier> PUBLIC FILE

This message is typed after the MCP performs an operation specified by a PUBLIC control card, providing that the SECMSG option (OPTN 29) is SET.

<file specifier> UNLOCKED

This message is typed after the MCP performs an operation specified by an UNLOCK control card, providing that the SECMSG option (OPTN 29) is SET.

APPENDIX F
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
000	BLOCK	Declaration not followed by semi- colon
001	BLOCK,ENTRY	Identifier declared twice in same block
002	PROCEDUREDEC,ENTRY	Specification PART contains iden- tifier not in format parameter PART.
003	BLOCK,ENTRY	Nonidentifier in identifier LIST of declaration
004	BLOCK	STREAM PROCEDURE Declaration pre- ceded by illegal declarator
005	BLOCK	PROCEDURE Declaration preceded by illegal declarator
006	BLOCK	PROCEDURE Identifier repeated in same block (not FORWARD)
007	BLOCK	PROCEDURE Identifier not followed by left parenthesis or semicolon in PROCEDURE Declaration
008	BLOCK	Formal parameter LIST not followed by right parenthesis
009	BLOCK	Formal parameter part not followed by semicolon
010	BLOCK	VALUE PART contains identifier not in formal parameter LIST

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
011	BLOCK	VALUE PART not ended by semicolon
012	BLOCK	Missing or illegal specification PART
013	ARRAE	OWN used in ARRAY Specification
014	ARRAE	SAVE used in ARRAY Specification
015	ENTRY	ARRAY call by value
016	ARRAE	ARRAY Identifier not followed by left bracket
017	ARRAE	Lower bound in ARRAY Declaration not followed by colon
018	ARRAE	Bound pair in ARRAY Declaration not followed by right bracket
019	ARRAE	Illegal lower bound designator in ARRAY Specification
020	BLOCK	OWN immediately before identifier (no type) in declaration
021	BLOCK	SAVE immediately before identifier (no type) in declaration
022	BLOCK	STREAM immediately before identifier (the word PROCEDURE left out)
023	CHKSQB	Declarator illegally preceded by another declarator
024	BLOCK	LABEL passed to a function

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
025	BLOCK,ENTER	Declarator or specifier illegally preceded by OWN, SAVE, or another declarator
026	IODEC	Missing left parenthesis in FILE Declaration
027	IODEC	Missing record size
028	IODEC	Illegal buffer part or SAVE Factor in FILE Declaration
029	IODEC	Missing right parenthesis in FILE Declaration
030	IODEC	Missing colon in disk description
031	BLOCK	Missing left parenthesis in LIST Declaration
032	FORMATGEN	Missing left parenthesis in FORMAT Declaration
033	IODEC,BLOCK	SWITCH Declaration does not have ← or FORWARD after identifier.
034	IODEC	Missing ← after SWITCH FILE Identifier
035	IODEC	NON-FILE Identifier in declaration of SWITCH FILE LIST
036	FORMATGEN	SWITCH FORMAT Identifier not followed by ←

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
037	FORMATGEN	Missing left parenthesis at start of SWITCH FORMAT LIST
038	FORMATGEN	SWITCH FORMAT Segment > 1022 words
039	BLOCK	Number of nested blocks > 31
040	IODEC	Program parameter block size exceeded
041	HANDLESWLIST	Missing ← after SWITCH LIST ID
042	HANDLESWLIST	Illegal list ID appearing in SWITCH LIST
043	IODEC	Missing right bracket after DISK in FILE Declaration
044	IODEC	Missing left bracket after DISK in FILE Declaration
045	DEFINEDEC,BLOCK	Missing = after defined identifier
046	ARRAE	Non-literal array bound not global to array declaration
047	TABLE	Item following @ not an integer
048	BLOCK	Number of parameters does not agree with number of parameters in FORWARD Declaration.
049	BLOCK	Type of this parameter does not agree with its type as given in FORWARD Declaration.

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
050	BLOCK	Value part differs from value part of FORWARD Declaration. Formal parameter of FORWARD Declaration and corresponding parameter in actual declaration are specified respectively as call-by-name and call-by-value, or vice versa.
059	ARRAE	Improper ARRAY size
060	FAULTSTMT	Missing ← in FAULT Statement
061	FAULTDEC	Invalid FAULT Type; must be FLAG, EXPOVR, ZERO, INTOVR, or INDEX
070	CASESTMT	Missing BEGIN
071	CASESTMT	Missing END
080	SEQCOMPARE	Sequence error
090	PARSE	Missing left bracket
091	PARSE	Missing colon
092	PARSE	Illegal bit number
093	PARSE	Field size must be literal
094	PARSE	Missing right bracket
095	PARSE	Illegal size
100	Anywhere	Undeclared identifier
101	CHECKER	Attempt has been made to address identifier which is local to one

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
		procedure and global to another. If quantity is procedure name or OWN variable, restriction is relaxed.
102	AEXP	Conditional expression not of arithmetic type
103	PRIMARY	Primary may not begin with this type quantity.
104	Anywhere	Missing right parenthesis
105	Anywhere	Missing left parenthesis
106	PRIMARY	Primary may not start with declarators.
107	BEXP	Expression not Boolean type
108	EXPRSS	Relation may not have condi- tional expressions as arithmetic expressions.
109	BOOSEC, SIMPBOO, BOOCOMP	Primary not Boolean type
110	BOOCOMP	Non-Boolean operator in Boolean expression
111	BOOPRIM	Expression (arithmetic, Boolean, or designational) may not begin with this type quantity.

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
112	BOOPRIM	Expression (arithmetic, Boolean, or designational) may not begin with declarator.
113	PARSE	Either syntax or range of literals for concatenate operator incorrect
114	DOTSYNTAX	Either syntax or range of literals for partial word designator incorrect
115	DEXP	Expression not of designational type
116	IFCLAUSE	Missing THEN
117	BANA	Missing left bracket
118	BANA	Missing right bracket
119	COMPOUNDTAIL	Missing semicolon or END
120	COMPOUNDTAIL	Missing END
121	ACTUALPARAPART	Indexed FILE may be passed by name only, and only to STREAM PROCEDURE. STREAM PROCEDURE may not RELEASE this type parameter.
122	ACTUALPARAPART	Expressions may not pass by name to STREAM PROCEDURES.
123	ACTUALPARAPART	Actual and formal parameters not same type

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
124	ACTUALPARAPART	Actual and formal arrays not same number of dimensions
125	ACTUALPARAPART	STREAM PROCEDURE may not be passed as an actual parameter to PROCEDURE.
126	ACTUALPARAPART	Actual parameter may not begin with this type quantity.
127	ACTUALPARAPART	This type quantity may not be passed to STREAM PROCEDURE.
128	ACTUALPARAPART	Actual and formal parameters do not agree in number. Extra right pa- renthesis
129	ACTUALPARAPART, IMPFUN	Illegal parameter delimiter
130	RELSESTMT	No FILE name
131	DOSTMT	Missing UNTIL
132	WHILESTMT	Missing DO
133	LABELR	Missing colon in LABEL
134	LABELR	LABEL not declared in this block
135	LABELR	LABEL has already occurred.
136	FORMATPHRASE, EXPLICITFORMAT	Improper FORMAT editing phrase
137	FORMATPHRASE, GETINT	FORMAT editing phrase does not have integer where required.

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGE

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
138	FORMATPHRASE, DIVDE	Width too small in E or F editing phrase
139	TABLE	DEFINE nested more than eight deep
140	NEXTENT	Integer in FORMAT > 1023
141	SCANNER, TABLE, FIXDEFINEINFO	Integer or identifier more than 63 characters
142	DEFINEGEN	DEFINE more than 2047 characters (blank suppressed)
143	COMPOUNDTAIL	Extra END
144	STMT	Statement may not start with this type identifier.
145	STMT	Statement may not start with this type quantity.
146	STMT	Statement may not start with decla- rator. (It may be missing END of PROCEDURE or misplaced declaration.)
147	SWITCHGEN	More than 256 expressions in SWITCH Declaration
148	GETSPACE	More than 1023 program reference table cells required for this pro- gram
149	GETSPACE	More than 255 stack cells required for this PROCEDURE

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
150	ACTUALPARAPART	Constants may not be passed by name to STREAM PROCEDURES.
151	FORSTMT	Index variable may not be Boolean.
152	FORSTMT	Missing ← following INDEX Variable
153	FORLIST	Missing UNTIL or WHILE in STEP Element
154	FORLIST	Missing DO in FOR Clause
155	IFEXP	Missing ELSE
156	LISTELEMENT	Designational expression may not be LIST Element.
157	LISTELEMENT	Row designator may not be LIST Element.
158	LISTELEMENT	Missing right bracket in elements
159	PROCSTMT	Illegal use of PROCEDURE or function identifier
160	PURGE	Declared LABEL did not occur.
161	PURGE	Declared FORWARD PROCEDURE did not occur.
162	PURGE	Declared FORWARD SWITCH did not occur.
163	EMITFORMAT	Width of field more than 63 characters

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
164	UNKNOWNSTMT	Missing comma in ZIP or WAIT Statement
165	IMPFUN	Missing comma in delay parameter list
172	DEFINEDEC	Too many parameters in parametric define
173	DEFINEDEC	Right parenthesis or right bracket expected after parameters in parametric define declaration
174	FIXDEFINEINFO	Incorrect number of parameters in parametric define invocation
199	E	INFO TABLE Array has overflowed.
200	EMIT,EMITWORD	Segment too large (> 4093 syllables)
201	VARIABLE	Partial word designator not leftmost in left part LIST
202	VARIABLE	Missing . or ←
203	VARIABLE	Wrong number of subscripts in row designator
204	VARIABLE	Missing right bracket in row designator
205	VARIABLE	Row designator outside of actual parameter in LIST or FILL Statement

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
206	VARIABLE	Missing right bracket
207	VARIABLE	Missing left bracket
208	VARIABLE	Wrong number of subscripts
209	VARIABLE	Partial word designator not left- most in left part LIST
210	VARIABLE	Missing . or ←
211	VARIABLE, DBLSTMT	PROCEDURE Identifier appears out- side of scope in left part.
212	VARIABLE	Sub-array designator permitted as actual parameter only
250	STREAMSTMT	Illegal STREAM Statement
251	STREAMSTMT	Missing ←
252	INDEXS	Missing + or -
253	INDEXS	Missing number or STREAM Variable
255	DSS	Missing string in DS← LT State- ment
256	RELEASES	Missing parenthesis, or FILE Identifier not a formal parameter
257	GOTOS, LABELS, JUMPS	LABEL specified not same nest level as preceding appearance of LABEL
258	LABELS	Missing colon

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
259	LABELS	LABEL appears more than once.
260	GOTOS	Missing LABEL in GO TO or JUMP OUT TO Statement
261	JUMPS	Missing OUT in JUMP OUT Statement
262	NESTS	Missing parenthesis
263	IFS	Missing SC in IF Statement
264	IFS	Missing relational in IF Statement
265	IFS	Missing ALPHA, DC, or string in IF Statement
266	IFS,RELEASES	Missing THEN in IF Statement
267	GOTOS,BLOCK	LABEL undefined in GO TO Statement
268	EMITC	Repeat index ≥ 64 specified; or too many formal parameters, locals, and labels
269	TABLE	Constant specified too large or too small
270	IFS	Relational operator other than = in test <source for ALPHA>
271	IFS	Improper construct for <source with literal>
281	DBLSTMT	Missing left parenthesis
282	DBLSTMT	Too many operators

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
283	DBLSTMT	Too many operands
284	DBLSTMT	Missing comma
285	DBLSTMT	Missing right parenthesis
286	DBLSTMT	Undeclared variable used
300	FILLSTMT	Identifier following FILL not ARRAY Identifier
301	FILLSTMT, MAKEALABEL	Missing WITH in FILL Statement
302	FILLSTMT	Improper FILL Element
303	FILLSTMT	Nonoctal character in octal FILL. The three low-order bits are con- verted and compilation continues.
304	FILLSTMT	Improper row designator
350	CHECKCOMMA	Missing or illegal parameter de- limiter in SORT or MERGE Statement
351	OUTPROCHECK	Illegal TYPE for SORT or MERGE Output Procedure
352	OUTPROCHECK	Output procedure in SORT or MERGE Statement does not have exactly two parameters.
353	OUTPROCHECK	First parameter of output procedure must be BOOLEAN.

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
354	OUTPROCHECK	Second parameter of output procedure must be ONE-DIMENSION ARRAY.
355	SORTSTMT	Missing left parenthesis
356	HVCHECK	Illegal TYPE for SORT or MERGE HIGH-VALUE Procedure
357	HVCHECK	HIGHVALUE Procedure does not have exactly one parameter.
358	HVCHECK	HIGHVALUE Procedure Parameter not ONE-DIMENSION ARRAY
359	EQLESCHECK	SORT or MERGE COMPARE Procedure not BOOLEAN
360	EQLESCHECK	COMPARE Procedure does not have exactly two parameters.
361	EQLESCHECK	COMPARE Procedure first parameter not ONE-DIMENSION ARRAY
362	EQLESCHECK	COMPARE Procedure second parameter not ONE-DIMENSION ARRAY
363	INPROCHECK	SORT Statement input procedure not BOOLEAN
364	INPROCHECK	Input procedure does not have exactly one parameter.
365	INPROCHECK	Input procedure parameter not ONE-DIMENSION ARRAY

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
366	SORTSTMT	Missing right parenthesis
367	MERGESTMT	Missing left parenthesis
368	MERGESTMT	More than seven or less than two files to merge
369	MERGESTMT	Missing right parenthesis
400	MERRIMAC	Missing FILE Identifier in MONITOR Declaration
401	MERRIMAC	Missing left parenthesis in MONITOR Declaration
402	MERRIMAC	Improper subscript for MONITOR LIST Element
403	MERRIMAC	Improper subscript expression de- limiter in MONITOR LIST Element
404	MERRIMAC	Improper number of subscripts in MONITOR LIST Element
405	MERRIMAC	LABEL or SWITCH monitored at improper level
406	MERRIMAC	Improper MONITOR LIST Element
407	MERRIMAC	Missing right parenthesis in MONITOR Declaration
408	MERRIMAC	Improper MONITOR Declaration Delimiter
409	DMUP	Missing FILE Identifier in DUMP Dec- laration

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
410	DMUP	Missing left parenthesis in DUMP Declaration
411	DMUP	Subscripted variable in DUMP LIST has wrong number of subscripts.
412	DMUP	Subscripted variable in DUMP LIST has wrong number of subscripts.
413	DMUP	Improper ARRAY DUMP LIST Element
414	DMUP	Illegal DUMP LIST Element
415	DMUP	More than 100 labels as DUMP LIST Elements in one DUMP Declaration
416	DMUP	Illegal DUMP LIST Element Delimiter
417	DMUP	Missing DUMP LABEL in DUMP Declaration
418	DMUP	Missing colon in DUMP Declaration
419	DMUP	Improper DUMP Declaration Delimiter
420	READSTMT	Missing left parenthesis in READ Statement
421	READSTMT	Missing left parenthesis in READ REVERSE Statement
422	READSTMT	Missing FILE in READ Statement
424	READSTMT	Improper FILE Delimiter in READ Statement

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
425	READSTMT	Improper FORMAT Delimiter in READ Statement
426	READSTMT, KLUDE	Improper delimiter for second parameter in READ Statement
427	READSTMT, KLUDE	Improper row designator in READ Statement
428	READSTMT, KLUDE	Improper row designator delimiter in READ Statement
429	READSTMT, KLUDE	Missing row designator in READ Statement
430	READSTMT	Improper delimiter preceding LIST in READ Statement
433	HANDLETHETAIL- ENDOFAREADOR- SPACESTATEMENT	Missing right bracket in READ or SPACE Statement
434	SPACESTMT	Missing left parenthesis in SPACE Statement
435	SPACESTMT	Improper FILE Identifier in SPACE Statement
436	SPACESTMT	Missing comma in SPACE Statement
437	SPACESTMT	Missing right parenthesis in SPACE Statement
438	WRITESTMT	Missing left parenthesis in WRITE Statement

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
439	WRITESTMT	Improper FILE Identifier in WRITE Statement
440	WRITESTMT	Improper delimiter for first parameter in WRITE Statement
441	WRITESTMT	Missing right bracket in <carriage control part> of WRITE Statement
442	WRITESTMT	Illegal carriage control delimiter in WRITE Statement
443	WRITESTMT	Improper second parameter delimiter in WRITE Statement
444	WRITESTMT	Improper row designator in WRITE Statement
445	WRITESTMT	Missing right parenthesis after row designator in WRITE Statement
446	WRITESTMT	Missing row designator in WRITE Statement
447	WRITESTMT	Improper delimiter preceding LIST in WRITE Statement
448	WRITESTMT	Improper LIST Delimiter in WRITE Statement
449	READSTMT	Improper LIST Delimiter in READ Statement
450	LOCKSTMT	Missing left parenthesis in LOCK Statement

APPENDIX F (cont)
 EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
451	LOCKSTMT	Improper FILE in LOCK Statement
452	LOCKSTMT	Missing comma in LOCK Statement
453	LOCKSTMT	Improper <unit disposition part> in LOCK Statement
454	LOCKSTMT	Missing right parenthesis in LOCK Statement
455	CLOSESTMT	Missing left parenthesis in CLOSE Statement
456	CLOSESTMT	Improper FILE in CLOSE Statement
457	CLOSESTMT	Missing comma in CLOSE Statement
458	CLOSESTMT	Improper <unit disposition part> in CLOSE Statement
459	CLOSESTMT	Missing right parenthesis in CLOSE Statement
460	RWNDSTMT	Missing left parenthesis in REWIND
461	RWNDSTMT	Improper <FILE part> IN REWIND Statement
462	RWNDSTMT	Missing right parenthesis in REWIND
463	BLOCK	MONITOR Declaration in specification of PROCEDURE
464	BLOCK	DUMP Declaration in specification of PROCEDURE

APPENDIX F (cont)
EXTENDED ALGOL SYNTACTICAL ERROR MESSAGES

<u>Error Message No.</u>	<u>Routine</u>	<u>Meaning</u>
465	DMUP	DUMP Indicator must be unsigned integer or simple variable.
500	SEARCHLIB	Illegal LIBRARY Identifier
501	SEARCHLIB	LIBRARY Identifier not in Directory
502	SEARCHLIB	Illegal LIBRARY start point
503	SEARCHLIB	Separator required between start point and length
504	SEARCHLIB	Illegal LIBRARY length
505	SEARCHLIB	Missing bracket
507*	SEARCHLIB	Magnetic tape positioning error
509	IODEC	Nonliteral FILE value not global to FILE declaration

* Although this is actually the result of a hardware malfunction, it is detected by the compiler and is therefore emitted as a syntax error message. The program does not compile properly from this point on, but compilation continues. The Subprogram Library Tape should be tried on a different unit.

APPENDIX G
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ACCESS MISSING

ACT. KEY MISSING

ACT. KEY QUALIFICATION ILLEGAL

ACTUAL KEY must be 77 level CMP or CMP-1 item.

ACT. KEY SIZE ILLEGAL

ACTUAL KEY > 11 digits

ACT. KEY TYPE ILLEGAL

ACTUAL KEY must be elementary.

ACT. KEY USAGE ILLEGAL

ACTUAL KEY must be COMPUTATIONAL or COMPUTATIONAL-1 elementary item
in WORKING-STORAGE.

ADD NO ELEMENT ITEMS

ARITHMETIC OPERAND CLASS xxxxx

Data-name xxxxx should be arithmetic operand, but its CLASS is
incorrect.

ARRAY SIZE ERROR STATEMENT TRUNCATION

Too many list elements in a diagnostic statement

ASSIGN SYNTAX ERROR

BUFFER MISSING

Disk file has been reserved declaring no alternate areas.

BY MISSING

Word BY missing in PERFORM Statement

CARD TRUNCATION xxxxx

CARDS nnnnn

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

nn CHARACTERS MISSING

Description does not match word boundary.

xxx CHARACTERS MISSING

COMPUTATIONAL item in record description is not word oriented, or record in file is not multiple of eight characters in length. Compiler inserts FILLER to make item start at beginning of word, thus changing record total size.

CHECK RECORD SIZE

This message caused by:

- a. RECORD SIZE declared different from SIZE in record description
- b. Character SIZE in BLOCK CONTAINS not integer multiple of record size

CLASS DECLARATION ERROR

Misplaced 77 level item

CLASS ERROR

In MEMORY SIZE clause, size not given as integer (should be given as number of words rather than number of memory modules)

CLASS ERROR

CLASS of item not declared with acceptable reserved word: NUMERIC, ALPHABETIC, ALPHANUMERIC, or AN

CLASS ERROR xxxxx

CLASS of data (xxxxx) either:

- a. Not numeric or edited numeric (arithmetic statement), or
- b. Invalid receiving field for MOVE Statement

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

CLASS ERROR ILLEGAL OPERAND

Operand in this statement has CLASS not legal in context.

COMPILE ERROR

If this message occurs after all other diagnostics have been removed, send SAR and source deck to Burroughs Corporation, Detroit.

COMPILE ERROR 01347000

Data-name has been defined more than 125 times.

COMPILE O.K. MO-DA-YR

Terminating message signifying successful compilation, as opposed to did-not compile. Certain warning messages, given by Compiler, may be shown without affecting compilation. This message does not imply that program is logically correct.

COMPILE TIME nnnnn SEC.

Information on compile time

CONDITIONAL GROUP SIGNED xxxxx

Comparison operand contains signed item in group identified by xxxxx.

CONDITIONAL GROUP SIZE xxxxx

Group identified by xxxxx contains item of variable size, or groups are of different sizes.

CONDITIONAL GROUP USAGE xxxxx

Group identified by xxxxx contains item of COMPUTATIONAL usage.

CONDITIONAL LITERAL OPERAND SIZE

Limit of 63 characters length exceeded

CONDITIONAL NAME ERROR

Condition-name must not be reserved word.

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

CONDITIONAL OPERAND CLASS ERROR

Numeric item is not allowed in this statement, or this is comparison of signed and unsigned items.

CONDITIONAL OPERAND SIGNED ERROR

Comparison operand is signed numeric.

CONDITIONAL OPERAND USAGE ERROR

Comparison of group items containing CMP items

CONDITIONAL SPECIFICATION SIZE ERROR

CONDITIONAL VALUE SIZE ERROR

88 level on item of more than 63 characters

COPY LEVEL ERROR

Level-number beyond range 0 through 49 because of incrementation during copy

(CORRESPONDING) xxxxx (DATA NAME) OF xxxxx (FILE NAME)

Itemizes all corresponding items being acted upon by corresponding operation

DECLARATION ERROR

RENAMING option used, but file-name not shown in prior SELECT Statement

DID NOT COMPILE MO-DA-YR.

DISK SIZE nnnnn

Requested information

DUPL. FILE NAME

Duplicated file-name in FD, MD, or SD entry

DUPL. \$\$ CARD

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

DUPLICATE LABEL

Name is duplicate of one previously defined. OBJECT-COMPUTER is specified two or more times (warning only)

ERRORS nnnnn

EXTRA ARITHMETIC OPERAND xxxxx

Several data-names shown following TO. One of data-names requires qualification.

EXTRA FILE DECLARATION ERROR

More than 50 files declared for file; and/or number of files times 18, plus length of all file-names in characters, plus 32 greater than 1024. (Note that assign to sort disk is two files and assign to sort disk and three sort-tapes is five files.)

FILE DECLARATION ERROR xxxxx

MONITOR or DUMP Statement does not declare file for printer.

FILE NOT SELECTED

Caused by:

- a. Compiler expecting word SELECT as next word
- b. File-name not shown in SELECT Statement in ENVIRONMENT DIVISION
- c. No files SELECTed

FROM MISSING PERFORM STATEMENT

Word FROM missing in PERFORM Statement

FROM MISSING xxxxx

GROUP CONDITIONAL OPERAND SIZE

Size of elementary item and group item in comparison is different, or one of items in comparison is variable-length.

APPENDIX F (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

GROUP CONDITIONAL OPERAND USAGE

COMPUTATIONAL ITEM contained in comparison between group and elementary item

GROUP PICTURE SPECIFICATION SYNTAX ERROR

PICTURE cannot be used at group level.

GROUP SIZE ERROR

Sum of SIZE of each elementary item does not agree with SIZE stated at group item level. Compiler continues, using sum of elementary item sizes, or record is not multiple of eight characters.

H-DATA-IMPROPER FOR TSS

HIERARCHY GROUP LEVEL ERROR

Level-number illegal; does not match previously defined group level-number

HYPHEN SPELLING ERROR

ILLEGAL ARITHMETIC CLASS xxxxx

Data-name shown does not have proper CLASS for use as arithmetic operand.

ILLEGAL ARITHMETIC LITERAL xxxxx

Literal shown is non-numeric and cannot be used on operand in arithmetic statement.

ILLEGAL ARITHMETIC OPERAND xxxxx

Caused by:

- a. Literal may only be preceded by + or -.
- b. Symbol should be plus or minus.
- c. Word is spelled incorrectly or used illegally.

ILLEGAL ASSIGNMENT FD xxxx

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL ASSIGNMENT MD xxxx

ILLEGAL ASSIGNMENT SD xxxx

File assignments in FILE-CONTROL Section of source program have been checked against file descriptions in DATA DIVISION, and conflict has been found. For example, file assigned to DISK has been described with FD in FILE Section rather than MD.

ILLEGAL ASSIGNMENT SPECIFICATION

ILLEGAL BLOCK SIZE SPECIFICATION

BLOCK SIZE for magnetic tape files specified greater than 1023 words

ILLEGAL CLASS DECLARATION

Sign has been specified for non-numeric field, or editing has been requested on non-numeric item.

ILLEGAL CLASS SIZE DEPENDING OPERAND

DEPENDING ON operand not unsigned integer; numeric field required

ILLEGAL CLASS SIZE DEPENDING OPERAND FILE xxxxx

Numeric item required

ILLEGAL CLASS SIZE DEPENDING OPERAND RECORD xxxxx

Numeric item required. No character count specified for TECHNIQUE-B or TECHNIQUE-C records

ILLEGAL CLASS SPECIFICATION

ILLEGAL COMPILE OPERATOR

Debugging compile MNEMONIC operator cannot be found.

ILLEGAL CONDITIONAL OPERAND

Caused by illegal Amount Comparison operand, or literal in condition having wrong CLASS

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL CONDITIONAL OPERAND xxxxx

Error in relation shown in xxxxx, relation incomplete, or xxxxx undefined

ILLEGAL CONDITIONAL OPERATOR xxxxx

Missing Relational Operator

ILLEGAL COPY

COPY of group item which includes this COPY entry

ILLEGAL DECLARATION

Numeric item JUSTIFIED LEFT must be integer; scaling not allowed

ILLEGAL DUPL. FILE NAME

File-names must be unique.

ILLEGAL DUPL. NAME xxxxx

Data-name given is duplicate, or xxxxx previously used as synonym.

ILLEGAL DUPL. SPECIFICATION

Item described within POINT LOCATION clause and PICTURE

ILLEGAL FILE INPUT-OUTPUT USAGE SPECIFICATION

More than one record per block in file declared as unblocked

ILLEGAL FILE NAME

File-name missing in SEEK

ILLEGAL FILE SIZE SPECIFICATION

Number of characters greater than 1023 words

ILLEGAL FILE TYPE xxxxx

Diagnostic statement refers to file with other than TECHNIQUE-A or unblocked records.

ILLEGAL FROM RECORD xxxxx

WRITE FROM can only be used on 01 level record.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL GO TO DEPENDING OPERAND

DEPENDING Operand must be integer.

ILLEGAL GROUP NAME xxxxx

Occurs because:

- a. Not legal to move elementary numeric or edited numeric item into group field
- b. xxxxx should not be group item for MOVE in process
- c. Group item appears in formula.

ILLEGAL GROUP OCCURS xxxxx

Group item with OCCURS in diagnostic statement

ILLEGAL INPUT-OUTPUT INTEGER

Reel number greater than three digits

ILLEGAL INPUT-OUTPUT SPECIFICATION

- a. INVALID KEY missing from WRITE to file assigned to disk
- b. Output file declared OPTIONAL
- c. TECHNIQUE missing when BLOCK CONTAINS > 1
- d. File-name or diagnostics missing
- e. OPEN OUTPUT 1 or CLOSE 1 on file assigned to disk

ILLEGAL INTEGER OPERAND

ILLEGAL INTEGER SORT SPECIFICATION

Clause RESERVE n ALTERNATE AREAS used with sort-file (SD entry), and n other than 1

ILLEGAL INTO RECORD xxxxx

Object of READ INTO clause must be an O1 record-name; xxxxx is not for O1 record-name.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL LABEL xxxxx

Paragraphs that may be ALTERed are those which contain only GO TO Statement. Subject paragraph not in that category

ILLEGAL LABEL OPERAND

Word beginning in column 8 of card not allowed there

ILLEGAL LABEL USAGE xxxxx

Label given is either reserved word, data-name, or non-unique or illegal reference to DECLARATIVES.

ILLEGAL LITERAL xxxxx

Receiving field in MOVE Statement cannot be literal.

ILLEGAL LITERAL CONDITIONAL xxxxx

Literal following ALL is non-integer numeric literal.

ILLEGAL MOVE OPERAND CLASS

Operand of MOVE attempts to place wrong CLASS of data in receiving field.

ILLEGAL MOVE OPERAND xxxxx

Improper MOVE made (e.g., ALPHABETIC, ALPHANUMERIC, or edited numeric field into numeric field)

ILLEGAL MOVE RECORD xxxxx

ILLEGAL MOVE USAGE OPERAND

ILLEGAL NAME xxxxx

Usually reserved word used incorrectly, data name > 30 characters, or integer used as data name

ILLEGAL OCCURS LEVEL

OCCURS clause used illegally at 01 level

ILLEGAL OCCURS USAGE xxxxx

Item in Diagnostic List is elementary OCCURS item.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL OCCURS VALUE DECLARATION

VALUE clause given for field requiring subscripting, or variable size DEPENDING ON option used with VALUE clause

ILLEGAL OPERAND NAME xxxx

Not proper item for arithmetic

ILLEGAL OPERAND xxxxx:

Data-name xxxxx:

- a. Not allowed in arithmetic statement.
- b. Other than elementary NUMERIC data item being varied in PERFORM Statement.
- c. Should be Figurative Constant other than END.
- d. Should be elementary item with DISPLAY USAGE.
- e. When xxxxx is), subscripting may be missing or incomplete.

ILLEGAL OPERAND WRITE STATEMENT

Integer associated with CHANNEL or LINES (e.g., nn LINES) not an unsigned integer or not NUMERIC data-name with unsigned integer value

ILLEGAL OPERATOR xxxxx

The xxxxx represents data-name in specification OUTPUT REVERSE.

ILLEGAL PICTURE SIZE DECLARATION

PICTURE specifies repetition of more than 127 occurrences of symbol.

ILLEGAL PICTURE SIZE SPECIFICATION

PICTURE greater than 120 characters

ILLEGAL PROCEDURE DIVISION MISSING END DEC

End declarative terminator not present

ILLEGAL PROCEDURE SPECIFICATION

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL PROGRAM IDENT

Actual program-id does not have quotes surrounding entry.

ILLEGAL QUALIFICATION

Incomplete, incorrect, or missing qualification. Word used as qualifier may be reserved word.

ILLEGAL QUALIFICATION xxxxx

When in synonym construct, synonym must be unique such that qualification never required nor allowed. Synonym shown not unique

ILLEGAL RECORD SIZE

Size of record exceeds 1023 words (8184 characters). 77 level > 1023 words

ILLEGAL RECORD SIZE xxxxx

Diagnostic Statement Record SIZE must be at least 15 words (120 characters).

ILLEGAL RECORD SIZE DECLARATION

SIZE of record > 1023 words

ILLEGAL RECORD SIZE SPECIFICATION

SIZE of record for magnetic tape exceeds limit of 1023 words (8184 characters); size is not 80 characters for punch.

ILLEGAL RECORD SPECIFICATION

ILLEGAL RENAMES OPERAND

Data-name given in RENAMES Statement does not appear in preceding record description.

ILLEGAL SIZE DECLARATION

Numeric item defined to have more than 63 integer places

ILLEGAL SIZE OPERAND xxxxx

In PERFORM Statement, data-name represented by xxxxx not allowed to have more than 11 characters

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL SIZE SPECIFICATION xxxxx

The xxxxx is double-precision field (more than 11 digits) and illegal in:

- a. COMPUTE Statement.
- b. Argument furnished to intrinsic function.
- c. Formula.

ILLEGAL SIZE/USAGE SPECIFICATION

COMPUTATIONAL item greater than 18 digits in length. Either SIZE or USAGE specification in error

ILLEGAL SORT FILE OPERATOR

ILLEGAL SORT INPUT-OUTPUT SPECIFICATION

Printed if TECHNIQUE was applied to sort-file. Warning-only message. TECHNIQUE ignored

ILLEGAL STATEMENT xxxxx

Word NEXT not followed by SENTENCE, or phrase TO PROCEED TO missing in ALTER Statement

ILLEGAL STATEMENT GROUP xxxxx

- a. Word SENTENCE does not follow NEXT, or is misspelled.
- b. Conditional statement must be followed by ELSE, OTHERWISE, or period.

ILLEGAL SUBSCRIPT COPY OPERAND

COPY Statement illegally refers to subscripted data-name.

ILLEGAL SUBSCRIPT MOVE OPERAND

- a. MOVE CORRESPONDING illegally refers to subscripted data-name.
- b. OPTION 3 of MOVE must not have subscripted operands.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

ILLEGAL SUBSCRIPT OPERAND

Either sending item or receiving item requires subscript.

ILLEGAL SYNTAX DECLARATION xxxxx

RENAMES entry not allowed as part of diagnostic statement

ILLEGAL TYPE xxxxx

Qualifier not group item or record-name

ILLEGAL USAGE OPERAND SIZE

ILLEGAL VALUE

Caused by:

- a. VALUE given not within allowable range or is improper
- b. VALUE may not be given for data-name in FILE SECTION,
or VALUE stated for label record field not allowed

ILLEGAL VALUE ASSIGNMENT xxxxx

Occurs in DUMP "label: data-name", but data-name not elementary
item or not numeric

ILLEGAL VALUE DECLARATION

SAVE FACTOR value illegal

ILLEGAL VALUE NAME xxxxx

CLASS of data-name xxxxx does not permit stated VALUE.

ILLEGAL WRITE NAME xxxxx

WRITE Statement must refer to 01 level record-name appearing in FILE
SECTION with FD description, not to SD record-name and not to record-
name appearing in WORKING-STORAGE or CONSTANT SECTIONS.

INPUT-OUTPUT MISSING xxxxx

Word INPUT or OUTPUT missing in USE Statement xxxxx.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

INTEGER CONDITIONAL OPERAND ERROR

Indicates:

- a. ALPHANUMERIC item not allowed
- b. Comparison of NUMERIC with non-numeric item; NUMERIC item not unsigned, integer, or of DISPLAY usage.

INTO MISSING

LABEL MISSING xxxxx

Paragraph following USE Statement does not contain label.

LEVEL ERROR

Used upon occurrence of one of following conditions:

- a. Level-number larger than 49 in record description
- b. 77 level-number in FILE SECTION
- c. 66 level-number not associated with RENAMES entry
- d. 77 level-number that appears after series of 77 level-numbers broken

LEVEL NOT RIGHT

Compiler malfunction. Please report details.

LIBRARY COPY SELECTED

COPY contains nested copy.

LIBRARY nnnn

LIBRARY READ ERROR

Error occurred in READ FROM LIBRARY.

LITERAL OPERATOR LITERAL ERROR

Statement indicates literal is compared with literal.

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

■ LITERAL SIZE ILLEGAL xxxxx

LITERAL xxxxx CHARACTERS

Non-numeric literal longer than 120 characters. Length xxx

■ LITERAL SPELLING ILLEGAL xxxxx

LITERAL SYNTAX PARENTHESIS

LITERAL TRUNCATION

Literal stated out of range of item

LITERAL VALUE NAME xxxxx

Value of xxxxx not proper in MOVE Statement, or not proper item for arithmetic statement

MEMORY SIZE nnnn

Requested information

■ MISSING ARITHMETIC OPERAND xxxxx

One of following conditions present:

- a. No receiving field following TO
- b. Only one operand shown
- c. Word xxxxx not proper in statement

MISSING ASSIGNMENT

SELECT clause should be followed by ASSIGN Clause.

■ MISSING ASSIGNMENT OPERATOR

FROM, n, or EQUALS missing in COMPUTE Statement

MISSING AT END READ STATEMENT

Either first READ Statement for file must have AT END explicitly given and all other READs in program should not have explicit AT END, or else every READ Statement for file in entire program must have explicit AT END Statement.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

MISSING CONDITIONAL OPERAND xxxxx
xxxxx not conditional operand

MISSING CONDITIONAL OPERATOR xxxxx
Relational operator should appear prior to xxxxx.

MISSING DECLARATION SECTION
Program does not contain section referred to by USE Statement in
DECLARATIVES, or misspelling caused it to appear to be missing.

MISSING DIVISION
Heading for division missing or misspelled.

MISSING END DEC ILLEGAL PROCEDURE DIVISION
End declarative terminator not present

MISSING FILE NAME
File-name must follow words INPUT or OUTPUT in OPEN Statement.

MISSING FILE SECTION SPECIFICATION xxxxx
Heading xxxxx appears instead of FILE SECTION.

MISSING FILE SIZE

MISSING FILE SPECIFICATION
Reference to BLOCK-COUNT, RECORD-COUNT, or REEL-NUMBER outside of
USE Procedure not qualified by file-name

MISSING GO TO PROCEDURE
ALTER refers to other than GO TO paragraph.

MISSING INPUT-OUTPUT OPERAND
Word INPUT or OUTPUT omitted from USE Statement

MISSING INPUT-OUTPUT SPECIFICATION
Invalid key clause missing in disk WRITE Statement, or verb OPEN not
followed by INPUT, OUTPUT, I-O, or INPUT-OUTPUT

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

MISSING LABEL

Label must identify first paragraph of section.

MISSING LEFT PARENTHESIS xxxxx

Left parenthesis is omitted:

- a. Instead of word xxxxx.
- b. Around argument for intrinsic function.
- c. Around diagnostic statement list.

MISSING OPERATOR xxxxx

Word BEFORE or AFTER not present in USE Statement

MISSING PARENTHESIS xxxxx

MISSING PERIOD

Required period missing

MISSING PERIOD xxxxx

Period is expected instead of name or symbol shown by xxxxx, or diagnostic statement does not end with period.

MISSING PROCEDURE DIVISION

PROCEDURE DIVISION heading omitted

MISSING PROGRAM IDENT

Non-numeric literal of program-id inside " " missing

MISSING QUALIFICATION

Word IN or OF omitted from qualification

MISSING QUALIFICATION xxxxx

Word xxxxx requires IN or OF as part of qualification.

MISSING QUALIFICATION NAME xxxxx

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

MISSING QUALIFICATION NAME

Necessary qualification is missing, or word used as qualifier cannot be found in program.

MISSING RECORD LEVEL

An 01 level record-name entry is omitted, or 01 level record-name does not begin record description following 77 level entries.

MISSING RECORD SIZE

Data name has appeared where compiler expected to find level-number.

MISSING RIGHT PARENTHESIS

Terminating parenthesis following synonym missing

MISSING RIGHT PARENTHESIS xxxxx

Right parenthesis should appear instead of xxxxx:

- a. At end of arithmetic expression in COMPUTE Statement.
- b. In conditional clause.
- c. Terminating list in diagnostic statement.

MISSING SECTION

Word SECTION missing from DATA DIVISION heading

MISSING SIZE DEPENDING DECLARATION

MISSING SIZE DEPENDING DECLARATION FILE xxxxx

MISSING SIZE DEPENDING DECLARATION RECORD xxxxx

File declared as TECHNIQUE-B or TECHNIQUE-C does not have variable-length data record.

MISSING SIZE SPECIFICATION

SIZE not specified for elementary item, or SIZE not specified for group item with VALUE Clause. Level-number missing on item following group item

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

MISSING STOP RUN STATEMENT

STOP RUN Statement does not appear in program, or it has been skipped because of NOTE.

MISSING SYNTAX OPERATOR xxxxx

Condition stated in DUMP Diagnostic Statement does not contain colon after xxxxx.

MISSING xxxxx READ STATEMENT

MISSING xxxxx WRITE STATEMENT

MONITOR STATEMENT MISSING

Statement OPEN OUTPUT DIAGNOSTIC or CLOSE DIAGNOSTIC appears, but Compiler did not find DUMP or MONITOR.

MOVE SYNTAX ERROR

Word following CORRESPONDING in MOVE Statement not proper, or non-unique data name (possibly synonym), used with CORRESPONDING option or word TO, missing in MOVE statement

MOVE TRUNCATION

Because of differences in description of items in MOVE Statement, truncation of digits will occur.

NEWTAPE nnnn

NO ELEMENT. ITEMS IN MOVE GROUP

MOVE CORRESPONDING Statement is given for which there are no corresponding elementary items. Level hierarchy of data-names referenced by CORRESPONDING option must match.

NO. SEGS. nnn

Information on number of segments

NOT FILE NAME xxxxx

Symbol shown in READ Statement not file-name

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

NOT RIGHT LEVEL

Compiler malfunction. Please report details.

NOT RECORD DECLARATION xxxxx

Symbol xxxxx shown as 01 record-name does not appear in DATA RECORDS clause in file description entry.

NOT RECORD NAME xxxxx

NOT SELECTED SORT TAPES

SD sort-file description file-name not subject of SELECT file-name ASSIGN TO n SORT-TAPES in ENVIRONMENT DIVISION (n is integer from 3 to 8)

OPERAND xxxxx NOT INTEGER

Data-name xxxxx not integer quantity for PERFORM Statement to execute integer TIMES

OPERAND RIGHT FILE RECORD xxxxx

OPERAND SIZE ERROR

DISPLAY Statement refers to data-name(s) whose total SIZE is greater than 176 characters.

OPTNL DECLARATION ERROR

Disk file may not be OPTIONAL.

PICTURE ERROR

PICTURE Specification is not proper, or number of symbols in PICTURE exceeds 30.

PICTURE PARENTHESIS USAGE ERROR

Number within parentheses specifying repetition not integer.

PICTURE SIZE ILLEGAL xxxxx

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

POSSIBLE ERROR RECORD SIZE

RELEASE Statement uses FROM Option, but size of two record areas not the same. Compiler will use shorter length of the two.

POSSIBLE MOVE CLASS ERROR

Items moved to different CLASS item

PROCEDURE MISSING xxxxx

USE Statement refers to label that is not part of DECLARATIVES.

PROCEDURE SIZE ERROR

Generated code for this procedure exceeds 1023 words in length. Additional dummy label should be added to procedure.

PROCEDURE xxxxx SIZE xxxxx

Information on procedure size

PRT nnn

Requested PRT number

PRT SIZE nnn

PRT SIZE ERROR

Caused by:

- a. Program Reference Table exceeds 511 words. Reduce number of 01 levels and COMP-1 items (in DATA DIVISION).
- b. Program Reference Table exceeds 1023 words. Reduce number of labels used in program (in PROCEDURE DIVISION).

QUOTE MISSING LITERAL GREATER THAN 120 CHARACTERS

READ STATEMENT SYNTAX ERROR

In READ Statement, word END missing, or SENTENCE in clause AT END GO TO NEXT SENTENCE missing

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

RECORD SIZE ERROR

Record size of sort-file not equal to record size of input or output file of sort. Program not compiled if record size less than sort record size

RECORD SIZE SELECTED; SORT VECTOR SIZE xxxxx

RECORD SIZE SELECTED xxxxx

SORT Statement information. Provided only when SPEC appears in \$ Card and MEMORY SIZE is below minimum (two times record size) for sort

REDEFINE ERROR

Operand of REDEFINES Clause illegal

REDEFINE SIZE ERROR

Area being redefined not equal to size of new description

REMOTE IMPROPER FOR NORMAL SYS

RIGHT PARENTHESIS MISSING

Synonym entry requires terminating right parenthesis.

RIGHT QUOTE MISSING

SEQUENCE ERROR

Sequence number appearing in card columns 1 through 6 not greater than number of preceding card. Message printed but compilation unharmed

SEQUENCE ERROR nnnn

SEQUENCE NUMBER TRUNCATION xxxxx

SIZE DECLARATION ERROR

Declared size of item and size shown by PICTURE not equal

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SIZE ERROR STATEMENT xxxxx

Word SIZE or ERROR missing from ON SIZE ERROR clause, or statement containing word or symbol shown by xxxxx has caused segment of code to exceed 1023 words in length. Additional labels should be added to reduce SIZE of segment.

SIZE ERROR WRITE STATEMENT

SIZE ILLEGAL ACT. KEY

SIZE ILLEGAL SIZE DEPENDING OPERAND

Variable size item in this statement has SIZE DEPENDING operand with size greater than 11.

SIZE ILLEGAL SIZE DEPENDING OPERAND FILE xxxxx

SIZE DEPENDING data-name referred to by READ Statement contains more than 11 decimal digits.

SIZE ILLEGAL SIZE DEPENDING OPERAND RECORD xxxxx

SIZE DEPENDING data-name referred to by WRITE Statement contains more than 11 decimal digits.

SIZE SPECIFICATION ERROR

Item declared as CMP-1 or COMPUTATIONAL-1 cannot exceed 11 decimal digits in length.

SORT MEMORY SIZE

Amount of memory used for sort if insufficient amount specified

SORT USAGE ERROR xxxxx

SORT VECTOR SIZE xxxxx

STATEMENT GROUP SIZE ERROR

Group size greater than 1023 words

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

STATEMENT TRUNCATION

Too many items in DUMP/MONITOR

SUBSCRIPT SPECIFICATION ERROR

Item declared as CMP-1 or COMPUTATIONAL-1 may not be subscripted.

SUBSCRIPT TRUNCATION

May not monitor item with more than 10 subscripts

SUBTR. NO ELEMENT. ITEMS

SYNTAX ERROR

In IDENTIFICATION DIVISION, word DIVISION missing following
IDENTIFICATION

Any following errors within ENVIRONMENT DIVISION cause this message:

- a. Omission of word DIVISION
- b. Omission of period
- c. Statement form wrong
- d. Invalid hardware-name
- e. No SOURCE-COMPUTER statement (warning only)
- f. MODS used instead of MOD in DISK SIZE

Any following errors within DATA DIVISION cause this message:

- a. Non-numeric literal specified when numeric literal needed
- b. Numeric literal specified when non-numeric literal needed
- c. VALUE Clause missing
- d. Format literal does not match item description.
- e. OCCURS Clause, with variable number of times, omitting
DEPENDING ON specification
- f. Word SECTION missing from headers

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

- g. FD or SD used incorrectly
- h. Reserved word used incorrectly
- i. Missing data-name
- j. Figurative constant used incorrectly
- k. Incorrect declaration, i.e., no level-number
- l. Item following SIZE (or SZ) not numeric
- m. Item following OCCURS not numeric

Any following errors within PROCEDURE DIVISION cause this message:

- a. Illegal operator in EXAMINE Statement
- b. Incorrect record-name (other than one defined by SD) included in RELEASE Statement.
- c. END-OF-JOB Card misplaced in source program
- d. Reserved word being EXAMINED
- e. READ or WRITE on SD File
- f. Preceding statement incomplete
- g. Tape opened I-0
- h. GO TO specifying data-name

SYNTAX ERROR xxxxx

xxxxxx in error

SYNTAX ERROR DIVISION MISSING

Word DIVISION missing or misspelled following PROCEDURE in heading

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR FILE DECLARATION xxxxx

File-name used is reserved word or has previously been used.

SYNTAX ERROR GO TO ERROR

Word after GO not TO

SYNTAX ERROR GO TO STATEMENT

Word TO does not follow GO TO statement, or DEPENDING ON clause missing

SYNTAX ERROR ILLEGAL SPELLING

SYNTAX ERROR LIBRARY MISSING

LIBRARY missing following FROM

SYNTAX ERROR MISSING FILE NAME

In CLOSE State, word following CLOSE not file-name

SYNTAX ERROR MISSING LABEL

Required label missing in statement, such as ALTER Label, TO PROCEED TO Label, or GO TO Label

SYNTAX ERROR MISSING LITERAL

STOP Statement not followed by reserved word RUN or by literal

SYNTAX ERROR MISSING PERIOD

Required period missing

SYNTAX ERROR MISSING PERIOD xxxxx>

Required period following xxxxx missing

SYNTAX ERROR MISSING QUALIFICATION xxxxx

xxxxx not valid qualifier, or MULTIPLY defined without qualifications

SYNTAX ERROR MISSING VERB

SYNTAX ERROR MOVE STATEMENT

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR NAME -DATA-

SYNTAX ERROR NAME -REMOTE-

SYNTAX ERROR PERFORM STATEMENT

SYNTAX ERROR READ STATEMENT

SYNTAX ERROR SORT STATEMENT 1

SORT Statement not first statement of paragraph, or attempt made to MONITOR paragraph which contains SORT Statement

SYNTAX ERROR SORT STATEMENT 2

Name of sort-file cannot be located in program. Probably because of misspelling

SYNTAX ERROR SORT STATEMENT 3

Word following SORT not file-name

SYNTAX ERROR SORT STATEMENT 4

File-name given, following SORT, is FD file description instead of SD sort-file description.

SYNTAX ERROR SORT STATEMENT 5

Wrong word appears following sort file-name. Normally, word is ON, ASCENDING, or DESCENDING.

SYNTAX ERROR SORT STATEMENT 6

Word following ON incorrect, possibly misspelled

SYNTAX ERROR SORT STATEMENT 7

Ordering of SORT Statement into ASCENDING or DESCENDING sequence not specified

SYNTAX ERROR SORT STATEMENT 8

More than 25 keys used in SORT Statement ordering

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR SORT STATEMENT 9

More than 25 keys used in SORT Statement ordering

SYNTAX ERROR SORT STATEMENT 10

Word ASCENDING or DESCENDING either missing or misspelled

SYNTAX ERROR SORT STATEMENT 11

One of key names given to ordering key cannot be located in program.

SYNTAX ERROR SORT STATEMENT 12

SORT Statement KEY data-name has USAGE that is neither DISPLAY nor COMPUTATIONAL. Caused by system failure of some type; either Master Control Program, COBOL Compiler, or hardware

SYNTAX ERROR SORT STATEMENT 13

CLASS of SORT Statement KEY data-name not correct. Caused by system failure within Master Control Program, COBOL Compiler, or hardware

SYNTAX ERROR SORT STATEMENT 14

SIGN of SORT Statement KEY data-name not correct. Caused by system failure within Master Control Program, COBOL Compiler, or hardware

SYNTAX ERROR SORT STATEMENT 15

SORT Statement KEY data-name requires subscripting that is not present.

SYNTAX ERROR SORT STATEMENT 16

Subscript for data-name in SORT Statement KEY is not unsigned integer quantity, and is illegal.

SYNTAX ERROR SORT STATEMENT 17

Closing parenthesis, following subscript list for SORT Statement KEY data-name, missing

SYNTAX ERROR SORT STATEMENT 19

Word following SORT Statement KEY data-name cannot be located in program, possibly because of misspelling.

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR SORT STATEMENT 20

Word following sort KEY not found in Dictionary because of misspelling, etc.

SYNTAX ERROR SORT STATEMENT 21

Reserved word, or symbol such as comma or right parenthesis, expected after one of SORT Statement KEY data-names, but not present, or unable to be identified because of misspelling, etc.

SYNTAX ERROR SORT STATEMENT 22

Word following INPUT cannot be properly identified.

SYNTAX ERROR SORT STATEMENT 23

Word following INPUT not PROCEDURE

SYNTAX ERROR SORT STATEMENT 24

SORT Statement does not contain INPUT PROCEDURE; therefore, USING file-name must be present, but USING cannot be located probably because of spelling error.

SYNTAX ERROR SORT STATEMENT 25

File-name following USING not file-name or is misspelled

SYNTAX ERROR SORT STATEMENT 26

File-name following USING cannot be identified as file-name.

SYNTAX ERROR SORT STATEMENT 27

File-name following USING has SD sort-file description entry instead of FD file description.

SYNTAX ERROR SORT STATEMENT 28

Word following USING file-name clause, or INPUT PROCEDURE, cannot be identified.

SYNTAX ERROR SORT STATEMENT 29

Word following OUTPUT not PROCEDURE

APPENDIX G (cont)

COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR SORT STATEMENT 30

OUTPUT PROCEDURE not specified in SORT Statement; therefore, GIVING file-name should be present

SYNTAX ERROR SORT STATEMENT 31

Data-name following GIVING cannot be identified in program.

SYNTAX ERROR SORT STATEMENT 32

Data-name following GIVING not file-name

SYNTAX ERROR SORT STATEMENT 33

Output file-name following GIVING described with SD sort-file description instead of FD file description

SYNTAX ERROR SORT STATEMENT 34

Period missing following SORT Statement. No other statement permitted within same sentence, or paragraph with SORT statement

SYNTAX ERROR SORT STATEMENT 35

Period terminating SORT Statement sentence missing

SYNTAX ERROR SORT STATEMENT 36

INPUT PROCEDURE and OUTPUT PROCEDURE both refer to same set of procedures. Illegal

SYNTAX ERROR SORT STATEMENT 37

SORT Statement attempting to use PRT locations in second half of PRT

SYNTAX ERROR SORT STATEMENT 38

Warning message, indicating more than one SORT Statement using same SD file as scratch tapes

SYNTAX ERROR SORT STATEMENT 39

Sort key not in sort record

SYNTAX ERROR SORT STATEMENT 41

One or more sort keys exceed 63 characters.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

SYNTAX ERROR VERB SYNTAX ERROR

SYNTAX ERROR WRITE STATEMENT
Illegal advancing operand

SYNTAX ERROR xxxxx STATEMENT

SYNTAX TYPE OPERAND

Improper use of reserved word in EXAMINE Statement, or literal
intended for SEARCH not bounded by quotes

TAPE-IN nnnnn

THIS PERFORM OPTION DELETED FROM TSS

TO MISSING xxxxx
xxxxx appears after EQUAL instead of TO

TOTAL SEG. SIZE nnnnn
Requested information

TYPE ILLEGAL ACT. KEY

UNIDENTIFIED ARITHMETIC NAME xxxxx
Data-name xxxxx cannot be located in program.

UNIDENTIFIED ARITHMETIC OPERAND xxxxx
Data-name or symbol cannot be located in program, probably because
of spelling errors.

UNIDENTIFIED COPY OPERAND
Data-name following COPY cannot be located in program thus far
because of spelling errors or forward reference.

UNIDENTIFIED HARDWARE
Hardware-name used not permitted in Compiler

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

UNIDENTIFIED LIBRARY NAME xxxxx

Name cannot be located in library.

UNIDENTIFIED NAME

Name given cannot be located in program.

UNIDENTIFIED NAME xxxxx

Data-name or label xxxxx cannot be located in program, or Compiler is looking for reserved word. RECORD or CHARACTER may be misspelled in BLOCK CONTAINS.

UNIDENTIFIED OPERAND xxxxx

Data given in forward reference not in DATA DIVISION

UNIDENTIFIED RECORD xxxxx

Record-name xxxxx, defined by 01 level entry, is not given in DATA RECORDS Clause; or record-name xxxxx, appearing in DATA RECORDS Clause, does not appear as an 01 level entry.

UNIDENTIFIED REDEFINE OPERAND

Operand does not appear in prior description.

UNIDENTIFIED VERB xxxxx

Verb beginning statement cannot be identified by Compiler.

-UNTIL- DELETED FROM TSS

USAGE ERROR

Caused by:

- a. COMPUTATIONAL usage has been declared for file which is to unit other than tape or drum. (If item is COMPUTATIONAL, it will be binary word.)
- b. Usage not declared as DISPLAY, COMPUTATIONAL, or CMP, or else omitted completely to imply DISPLAY
- c. Usage must be DISPLAY for item in EXAMINE Statement.

APPENDIX G (cont)
COBOL COMPILER ERROR AND DIAGNOSTIC MESSAGES

■ USAGE SPECIFICATION ERROR

VALUE NOT INTEGER xxxxx

Value stated in diagnostic dump statement, as condition when statement is to be executed, not an integer

VALUE TYPE ERROR

VALUE stated for level 88 entry does not agree with CLASS given for conditional-variable.

■ -WHEN- DELETED FROM TSS

APPENDIX H
FORTRAN COMPILER ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
000	Syntax error.
001	Missing operator or punctuation.
002	Conflicting COMMON and/or EQUIVALENCE ALLOCATION.
003	Missing right parenthesis.
004	ENTRY statement illegal in main program or BLOCK DATA.
005	Missing END statement.
006	Arithmetic expression required.
007	Logical expression required.
008	Too many left parentheses.
009	Too many right parentheses.
010	Formal parameter illegal in COMMON.
011	Formal parameter illegal in EQUIVALENCE.
012	This statement illegal in BLOCK DATA subprogram.
013	INFO array overflow.
014	Improper DO nest.
015	DO label previously defined.
016	Unrecognized statement type.
017	Illegal DO statement.
018	FORMAT statement must have label.
019	Undefined label.
020	Multiple definition.
021	Illegal identifier class in this context.
022	Unpaired quotes in FORMAT.
023	Not enough subscripts.
024	Too many subscripts.
025	FUNCTION or SUBROUTINE previously defined.
026	Formal parameter multiply defined in heading.
027	Illegal use of NAMELIST.
028	Number of parameters inconsistent.
029	Cannot branch to FORMAT statement.
030	SUBROUTINE or FUNCTION not defined in program.
031	Identifier already given type.
032	Illegal FORMAT syntax.

APPENDIX H (cont)
FORTRAN COMPILER ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
033	Incorrect use of file.
034	Inconsistent use of identifier.
035	Array identifier expected.
036	Expression value required.
037	Illegal file card syntax.
038	Illegal control element.
039	Declaration must precede first reference.
040	Inconsistent use of label as parameter.
041	Number of parameters disagrees with previous reference.
042	Illegal use of formal parameter.
043	Error in hollerith literal character count.
044	Illegal use of formal parameter.
045	Too many segments in source program.
046	Too many PRT assignments in source program.
047	Last block declaration had less than 1024 words.
048	Illegal I/O list element.
049	Left side must be simple or subscripted variable.
050	Variable expected.
051	Illegal use of .OR.
052	Illegal use of .AND.
053	Illegal use of .NOT.
054	Illegal use of relational operator.
055	Illegal mixed types.
056	Illegal expression structure.
057	Illegal parameter.
058	Record block greater than 1023.
059	Too many optional files.
060	File cards must precede source deck.
061	Binary WRITE statement has no list.
062	Undefined FORMAT number.
063	Illegal exponent in constant.
064	Illegal constant in DATA statement.
065	Main program missing.

APPENDIX H (cont)
FORTRAN COMPILER ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
066	Parameter must be array identifier.
067	Parameter must be expression.
068	Parameter must be label.
069	Parameter must be FUNCTION identifier.
070	Parameter must be FUNCTION or SUBROUTINE ID.
071	Parameter must be SUBROUTINE identifier.
072	Parameter must be array identifier or expression.
073	Arithmetic - logical conflict on store.
074	Array ID must be subscripted in this context.
075	More than one main program.
076	Only COMMON elements permitted.
077	Too many files.
078	FORMAT or NAMELIST too long.
079	Formal parameter must be array identifier.
080	Formal parameter must be simple variable.
081	Formal parameter must be FUNCTION identifier.
082	Formal parameter must be SUBROUTINE identifier.
083	Formal parameter must be FUNCTION or SUBROUTINE.
084	DO or implied DO index must be integer or real.
085	Illegal complex constant.
086	Illegal mixed type store.
087	Constant exceeds hardware limits.
088	Parameter type conflicts with previous use.
089	Complex expression illegal in IF statement.
090	Complex expression illegal in relation.
091	Too many formats referenced but not yet found.
092	Variable array bound must be formal variable.
093	Array bound must have integer or real type.
094	Comma or right parenthesis expected.
095	Array already given bounds.
096	Only formal arrays must be given variable bounds.
097	Missing left parenthesis in implied DO.
098	Subscript must be integer or real.

APPENDIX H (cont)
FORTRAN COMPILER ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
099	Array size cannot exceed 32767 words.
100	COMMON or EQUIVALENCE block cannot exceed 32767 words.
101	This statement illegal in logical IF.
102	Real or integer type required.
103	Array bound information required.
104	Replacement operator expected.
105	Identifier expected.
106	Left parenthesis expected.
107	Illegal formal parameter.
108	Right parenthesis expected.
109	Statement number expected.
110	Slash expected.
111	ENTRY statement cannot start program unit.
112	Array must be dimensioned prior to equivalence statement.
113	Integer constant expected.
114	Comma expected.
115	Slash or end of statement expected.
116	FORMAT, array or NAMELIST expected.
117	End of statement expected.
118	IO statement with NAMELIST cannot have IO list.
119	Comma or end of statement expected.
120	String too long.
121	Missing quote at end of string.
122	Illegal array bound.
123	Too many hanging branches.
124	Too many COMMON or EQUIVALENCE elements.
125	Asterisk expected.
126	Comma or slash expected.
127	Data set too large.
128	Too many ENTRY statements in this subprogram.
129	Decimal width exceeds field width.
130	Unspecified field width.
131	Unspecified scale factor.

APPENDIX H (cont)
FORTRAN COMPILER ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
132	Illegal FORMAT character.
133	Unspecified decimal field.
134	Decimal field illegal for this specifier.
135	Illegal label.
136	Undefined NAMELIST.
137	Multiply defined action labels.
138	Too many nested DO statements.
139	Statement FUNCTION ID and expression disagree in type.
140	Illegal use of statement FUNCTION.
141	Unrecognized construct.
142	RETURN, STOP or CALL EXIT required in subprogram.
143	Format number used previously as label.
144	Label used previously as FORMAT number.
145	Non-standard RETURN requires label parameters.
146	Double or complex requires even offset.
147	FORMAT parameter illegal in DATA statement.
-	Sequence error "n" < "p", where n is the new sequence number and p is the old sequence number.

APPENDIX I
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
01	The identifier is not a real or integer, single precision variable.
02	The statement number is negative or zero.
03	The statement number contains more than five digits.
04	A special character is incorrectly placed in the statement.
05	An identifier contains more than six characters.
06	An identifier is missing.
07	The first character of an identifier is not alphabetic.
08	An array contains a zero subscript.
09	The identifier is not a simple variable.
10	The maximum limit of an array subscript exceeds 32767.
11	The number of subscripts of a subscripted variable does not agree with the number declared by the DIMENSION statement.
12	A subscript of an identifier is in error.
13	The statement is incomplete.
14	The number of actual arguments of a function does not agree with the number of dummy arguments.
15	The statement contains an undefined function.
16	An identifier used to represent a function has not been defined as such.

APPENDIX I (cont)
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
17	The identifier is not an integer, real or logical, single precision variable.
18	An identifier is used improperly in the statement.
19	The subroutine name in the CALL statement has already been used to represent another identifier.
20	There is an illegal identifier in a SUBROUTINE argument list.
21	The expression contains an illegally placed character.
22	The relational expression does not contain a relational operator.
23	The FORMAT statement calls for a total field width count exceeding 132 print positions.
24	The tape unit number is zero or exceeds the maximum limit.
25	An illegal identifier is present in an I/O list or an assignment statement.
26	An equal sign is missing from a compound I/O list.
27	A right parenthesis is missing from a compound I/O list.
28	A comma is missing from a compound I/O list.
29	A compound I/O list contains duplicate index identifiers.
30	The number of nested I/O implied DO loops exceeds the maximum allowed.
31	A compound I/O list is incorrectly written.

APPENDIX I (cont)
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
32	An I/O list contains an identifier name that does not begin with an alphabetic character.
33	A MONITOR or DUMP statement contains an illegal element.
34	The array has been previously dimensioned.
35	An array has more than three dimensions.
36	A variable name used as a dimension is not present in the subprogram argument list.
37	A variable name used as a dimension is not a simple integer.
38	A DO loop overlays another DO loop.
39	A FORMAT statement does not have a statement number.
40	The name of a statement function has been used to represent another type identifier.
41	The subprogram argument list contains duplicate arguments or an argument list that is represented by the same identifier as the subprogram.
42	A DO loop ends with another DO statement.
43	The terminal statement number of a DO loop precedes the DO statement.
44	An index identifier of a DO statement is not an integer variable.
46	An illegal identifier appears in a COMMON or EQUIVALENCE list.

APPENDIX I (cont)
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
47	The number of arguments in the argument list of a SUBROUTINE call does not agree with the number of dummy arguments.
48	This statement is not implemented.
49	A SUBROUTINE identifier exceeds six characters.
50	An identifier appears more than once in COMMON.
51	A dummy argument of a subprogram appears in the EQUIVALENCE list.
52	Two identifiers in COMMON have been set equivalent.
53	An identifier is missing from an EQUIVALENCE list element.
54	An identifier in an EXTERNAL statement is present in COMMON or in an EQUIVALENCE list.
55	An identifier is defined before its appearance in a Type statement.
56	The statement is undefined.
57	The / is used illegally.
58	The \$ is used illegally.
59	The first special character in the statement is in error.
60	The statement number has been duplicated.
61	The number of DATA elements in the DATA statement does not agree with the number of identifiers in the DATA list.
62	A DATA list identifier is not a variable name.

APPENDIX I (cont)
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
63	An * in a DATA statement is preceded by a zero.
64	The identifier is not a logical variable.
65	The character following the @ is in error.
66	A MAX or MIN function has less than two arguments.
67	The DATA statement contains an illegal logical constant.
68	The number of characters in the statement exceeds the maximum allowed.
69	The TAPE declaration on the START\$ card specifies an excessive number of units.
70	The integer following the OCTAL option on the START\$ card exceeds 1023 or has been declared zero.
71	The START\$ card contains an illegal item.
72	The final \$ is missing on the START\$ card.
74	The statement function has been defined after the first executable statement.
75	An identifier other than a subscripted variable is dimensioned.
76	A file number ≤ 0 has been given for a disk file.
77	The number of areas given for a disk file is ≤ 0 or > 20 .
78	The number of logical records for a disk file is ≤ 0 or > 1048575 .
79	The number of words in a record for a disk file is ≤ 0 or > 524287 .

APPENDIX I (cont)
FORTRAN TRANSLATOR ERROR MESSAGES

<u>Error Message No.</u>	<u>Meaning</u>
80	The identifier representing an associated variable for a disk file exceeds six characters.
81	The size of the record has not been given for the temporary file.
82	The associated variable has not been specified for the random file.
83	A Find statement is not necessary and is not implemented.
84	The Define File statement is not necessary and is not implemented.
85	There is not enough information for the disk file.
86	The save factor for a disk file is ≤ 0 or > 1023 .
90	A FORTRAN II logical expression contains an illegal symbol.
91	A FORTRAN II logical expression uses a (-) incorrectly.
92	A FORTRAN II logical expression is not assigned to a REAL name.
93	A FORTRAN II IF statement contains an illegal two way branch.
94	The TWO/FOUR indicator card is missing.
95	An identifier name in FORTRAN II exceeds three characters and ends in F.

APPENDIX J
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
000	BLOCK	DECLARATION NOT FOLLOWED BY SEMICOLON.
001	BLOCK,ENTRY	IDENTIFIER DECLARED TWICE IN SAME BLOCK.
002	PROCEDUREDEC, ENTRY	SPECIFICATION PART CONTAINS IDENTIFIER NOT APPEARING IN FORMAT PARAMETER PART.
003	BLOCK,ENTRY	NON-IDENTIFIER APPEARS IN IDENTIFIER LIST OF DECLARATION.
005	BLOCK	PROCEDURE DECLARATION PRECEDED BY ILLEGAL DECLARATOR.
006	BLOCK	PROCEDURE IDENTIFIER USED BEFORE IN SAME BLOCK (NOT FORWARD).
007	BLOCK	PROCEDURE IDENTIFIER NOT FOLLOWED BY (OR SEMICOLON IN PROCEDURE DECLARATION.
008	BLOCK	FORMAL PARAMETER LIST NOT FOLLOWED BY).
009	BLOCK	FORMAL PARAMETER PART NOT FOLLOWED BY SEMICOLON.
010	BLOCK	VALUE PART CONTAINS IDENTIFIER WHICH DID NOT APPEAR IN FORMAL PARAPART.
011	BLOCK	VALUE PART NOT ENDED BY SEMICOLON.
012	BLOCK	MISSING OR ILLEGAL SPECIFICATION PART.
013	ARRAE	OWN USED IN ARRAY SPECIFICATION.
014	ARRAE	SAVE USED IN ARRAY SPECIFICATION.
015	ENTRY	ARRAY CALL-BY-VALUE NOT IMPLEMENTED.
016	ARRAE	ARRAY ID IN DECLARATION NOT FOLLOWED BY [.
017	ARRAE	LOWER BOUND IN ARRAY DEC NOT FOLLOWED BY :.
018	ARRAE	BOUND PAIR LIST NOT FOLLOWED BY] .
019	ARRAE	ILLEGAL LOWER BOUND DESIGNATOR IN ARRAY SPECIFICATION.
020	BLOCK	OWN APPEARS IMMEDIATELY BEFORE IDENTIFIER (NO TYPE).
021	BLOCK	SAVE APPEARS IMMEDIATELY BEFORE IDENTIFIER (NO TYPE).
023	CHKSOB	DECLARATOR PRECEDED ILLEGALLY BY ANOTHER DECLARATOR.

APPENDIX J (cont)

COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
024	BLOCK	LABEL CANNOT BE PASSED TO FUNCTION.
025	BLOCK, ENTER	DECLARATOR OR SPECIFIER ILLEGALLY PRECEDED BY OWN OR SAVE OR SOME OTHER DECLARATOR.
026	IODEC	MISSING (IN FILE DEC.
027	IODEC	MISSING RECORD SIZE.
028	IODEC	ILLEGAL BUFFER PART OR SAVE FACTOR IN FILE DEC.
029	IODEC	MISSING) IN FILE DEC.
030	IODEC	MISSING COLON IN DISK DESCRIPTION.
031	BLOCK	MISSING (IN LISTDEC.
032	FORMATGEN	MISSING (IN FORMAT DEC.
033	IODEC, BLOCK	SWITCH DEC DOES NOT HAVE ← OR FORWARD AFTER IDENTIFIER.
034	IODEC	MISSING ← AFTER FILED.
035	IODEC	NON FILE ID APPEARING IN DECLARATION OF SWITCHFILE.
036	FORMATGEN	FORMAT ID NOT FOLLOWED BY ←.
037	FORMATGEN	MISSING (AT START OF FORMATPHRASE.
038	FORMATGEN	FORMAT SEGMENT > 1023 WORDS.
039	BLOCK	NUMBER OF NESTED BLOCKS IS GREATER THAN 31.
040	IODEC	PROGRAM PARAMETER BLOCK SIZE EXCEEDED.
041	HANDLESWLIST	MISSING ← AFTER SWITCH LIST ID.
042	HANDLESWLIST	ILLEGAL LIST ID APPEARING IN SWITCH LIST
043	IODEC	MISSING] AFTER DISK IN FILEDEC.
044	IODEC	MISSING [AFTER DISK IN FILEDEC.
045	DEFINDEC, BLOCK	MISSING "=" AFTER DEFINE ID.
046	ARRAE	NON-LITERAL ARRAY BOUND NOT GLOBAL TO ARRAY DECL.
047	TABLE	ITEM FOLLOWING @ NOT A NUMBER.
048	BLOCK	NUMBER OF PARAMETERS DIFFERS FROM FWD DECL.
049	BLOCK	CLASS OF PARAMETER DIFFERS FROM FWD DECL.

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
050	BLOCK	VALUE PART DIFFERS FROM FWD DECL.
059	ARRAE	MISSING ← IN FAULT STATEMENT.
061	FAULTDEC	INVALID FAULT TYPE: MUST BE FLAG, EXPOVR, ZERO, INTOVR, OR INDEX.
062	SCANSTMT OR REPLACESTMT	LEVEL OF POINTER EXPRESSION EXCEEDS LEVEL OF UPDATE POINTER IDENTIFIER.
063	SCANSTMT OR REPLACESTMT	UPDATE POINTER MAY NOT BE CALL-BY-NAME FORMAL PARAMETER.
070	CASESTMT	MISSING "BEGIN".
071	CASESTMT	MISSING END.
072	SCANSTMT OR REPLACESTMT	POINTER IDENTIFIER REQUIRED.
073	SCANSTMT OR REPLACESTMT	SIMPLE ARITHMETIC VARIABLE REQ.
074	SCANSTMT OR REPLACESTMT	RELATIONAL OP OR IN EXPECTED.
075	SCANSTMT OR REPLACESTMT	CONDITION MUST START WITH WHILE OR UNTIL.
076	REPLACESTMT	BY MISSING AFTER DESTINATION POINTER.
077	REPLACESTMT	SOURCE MUST BE POINTER OR ARITHMETIC EXP.
078	SCANSTMT OR REPLACESTMT	ALPHA REQUIRED AFTER IN.
079	PRIMARY	ILLEGAL EXPRESSION TYPE.
080	PRIMARY	MISSING COMMA.
090	PARSE	MISSING LEFT BRACKET.
091	PARSE	MISSING COLON.
092	PARSE	ILLEGAL BIT NUMBER.
093	PARSE	FIELD SIZE MUST BE LITERAL.
094	PARSE	MISSING RIGHT BRACKET.
095	PARSE	ILLEGAL FIELD SIZE.
100	Anywhere	UNDECLARED IDENTIFIER.
101	CHECKER	AN ATTEMPT HAS BEEN MADE TO ADDRESS AN IDENTIFIER WHICH IS LOCAL TO ONE PROCEDURE AND GLOBAL TO ANOTHER. IF THE QUANTITY IS A PROCEDURE NAME OR AN OWN VARIABLE, THIS RESTRICTION IS RELAXED.

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
102	AEXP	CONDITIONAL EXPRESSION IS NOT OF ARITHMETIC TYPEH.
103	PRIMARY	PRIMARY MAY NOT START WITH A QUANTITY OF THIS TYPE.
104	Anywhere	MISSING RIGHT PARENTHESIS.
105	Anywhere	MISSING LEFT PARENTHESIS.
106	PRIMARY	PRIMARY MAY NOT START WITH DECLARATOR.
107	BEXP	THE EXPRESSION IS NOT OF BOOLEAN TYPE.
108	EXPRSS	A RELATION MAY NOT HAVE CONDITIONAL EXPRESSIONS AS THE ARITHMETIC EXPRESSIONS.
109	BOOSEC, SIMPBOO, AND BOOCOMP	THE PRIMARY IS NOT BOOLEAN.
110	BOOCOMP	A NON-BOOLEAN OPERATOR OCCURS IN A BOOLEAN EXPRESSION.
111	BOOPRIM	NO EXPRESSION (ARITHMETIC, BOOLEAN, OR DESIGNATIONAL) MAY START WITH A QUANTITY OF THIS TYPE.
112	BOOPRIM	NO EXPRESSION (ARITHMETIC, BOOLEAN, OR DESIGNATIONAL) MAY START WITH A DECLARATOR.
113	PARSE	EITHER THE SYNTAX OR THE RANGE OF THE LITERALS FOR A CONCATENATE OPERATOR IS INCORRECT.
114	DOTSYNTAX	EITHER THE SYNTAX OR THE RANGE OF THE LITERALS FOR A PARTIAL WORD DESIGNATOR IS INCORRECT.
115	DEXP	THE EXPRESSION IS NOT OF DESIGNATIONAL TYPE.
116	IFCLAUSE	MISSING THEN.
117	BANA	MISSING LEFT BRACKET.
118	BANA	MISSING RIGHT BRACKET.
119	COMPOUNDTAIL	MISSING SEMICOLON OR END.
120	COMPOUNDTAIL	MISSING END.
121	ACTUALPARAPART	INDEXED FILES MAY NOT BE PASSED.
123	ACTUALPARAPART	THE ACTUAL AND FORMAL PARAMETERS DO NOT AGREE AS TO TYPE.
124	ACTUALPARAPART	ACTUAL AND FORMAL ARRAYS DO NOT HAVE SAME NUMBER OF DIMENSIONS.

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
126	ACTUALPARAPART	NO ACTUAL PARAMETER MAY START WITH A QUANTITY OF THIS TYPE.
128	ACTUALPARAPART, PROCSTMT	EITHER ACTUAL AND FORMAL PARAMETERS DO NOT AGREE AS TO NUMBER, OR EXTRA RIGHT PARENTHESIS.
129	ACTUALPARAPART, IMPFUN	ILLEGAL PARAMETER DELIMITER.
130	RELSESTMT	NO FILE NAME.
131	DOSTMT	MISSING UNTIL.
132	WHILESTMT	MISSING DO.
133	LABELR	MISSING COLON.
134	LABELR	THE LABEL WAS NOT DECLARED IN THIS BLOCK.
135	LABELR	THE LABEL HAS ALREADY OCCURRED.
136	FORMATPHRASE	IMPROPER FORMAT EDITING PHRASE.
137	FORMATPHRASE, GETINT	A FORMAT EDITING PHRASE DOES NOT HAVE AN INTEGER WHERE AN INTEGER IS REQUIRED.
138	FORMATPHRASE, DIVIDE	THE WIDTH IS TOO SMALL IN E OR F EDITING PHRASE.
139	TABLE	DEFINE IS NESTED MORE THAN EIGHT DEEP.
140	NEXTENT	AN INTEGER IN A FORMAT IS GREATER THAN 1023.
141	SCANNER, TABLE, FIXDEFINEINFO	INTEGER OR IDENTIFIER HAS MORE THAN 63 CHARACTERS.
142	DEFINEGEN	A DEFINE CONTAINS MORE THAN 2047 CHARACTERS (BLANK SUPPRESSED).
143	COMPOUNDTAIL	EXTRA END.
144	STMT	NO STATEMENT MAY START WITH THIS TYPE IDENTIFIER.
145	STMT	NO STATEMENT MAY START WITH THIS TYPE QUANTITY.
146	STMT	NO STATEMENT MAY START WITH A DECLARATOR - MAY BE A MISSING END OF A PROCEDURE OR A MISPLACED DECLARATION.
147	SWITCHGEN	MORE THAN 256 EXPRESSIONS IN A SWITCH DECLARATION.
148	GETSPACE	MORE THAN 1023 PROGRAM REFERENCE TABLE CELLS ARE REQUIRED FOR THIS PROGRAM.
149	GETSPACE	MORE THAN 255 STACK CELLS ARE REQUIRED FOR THIS PROCEDURE.

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
150	THRUSTMT	MISSING DO IN THRU CLAUSE.
151	FORSTMT	INDEX VARIABLE MAY NOT BE BOOLEAN.
152	FORSTMT	MISSING LEFT ARROW FOLLOWING INDEX VARIABLE.
153	FORLIST	MISSING UNTIL OR WHILE IN STEP ELEMENT.
154	FORLIST	MISSING DO IN FOR CLAUSE.
155	IFEXP	MISSING ELSE.
156	LISTELEMENT	A DESIGNATIONAL EXPRESSION MAY NOT BE A LIST ELEMENT.
157	LISTELEMENT	A ROW DESIGNATOR MAY NOT BE A LIST ELEMENT.
158	LISTELEMENT	MISSING RIGHT BRACKET IN GROUP ELEMENTS.
159	PROCSTMT	ILLEGAL USE OF PROCEDURE OR FUNCTION IDENTIFIER.
160	PURGE	DECLARED LABEL DOES NOT OCCUR.
161	PURGE	DECLARED FORWARD PROCEDURE DOES NOT OCCUR.
162	PURGE	DECLARED SWITCH FORWARD DOES NOT OCCUR.
163	EMITFORMAT	THE WIDTH OF A FIELD IS MORE THAN 63.
164	UNKNOWNSTMT	MISSING COMMA IN ZIP OR WAIT STATEMENT.
165	IMPFUN	MISSING COMMA IN DELAY PARAMETER LIST.
166	PEXP	THE EXPRESSION IS NOT OF POINTER TYPE.
167	PTRPRIMARY	POINTER PRIMARY MAY NOT START WITH A QUANTITY OF THIS TYPE.
168	VARIABLE	POINTER MAY NOT HAVE PARTIAL WORD SYNTAX.
169	ARRAE	POINTER ARRAYS NOT PERMITTED.
170	SWAPSTMT	MISSING COMMA.
171	SWAPSTMT	PARAMETERS MUST BE 2-DIMENSIONAL ARRAYS.
172	DEFINEDEC	TOO MANY PARAMETERS IN PARAMETRIC DEFINE.
173	DEFINEDEC	RIGHT PARENTHESIS OR RIGHT BRACKET EXPECTED AFTER PARAMETER IN PARAMETRIC DEFINE DECLARATION.
174	FIXDEFINEINFO	INCORRECT NUMBER OF PARAMETERS IN PARAMETRIC DEFINE INVOCATION.
199	E	INFO "TABLE" ARRAY HAS OVERFLOWED.
200	EMIT,EMITWORD	SEGMENT TOO LARGE (> 4093 SYLLABLES).

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
201	VARIABLE	VARIABLES: PARTIAL WORD DESIGNATOR NOT LEFT-MOST IN LEFT PART LIST.
202	VARIABLE	VARIABLES: MISSING . OR ← .
203	VARIABLE	WRONG NUMBER OF SUBSCRIPTS IN A ROW DESIGNATOR.
204	VARIABLE	MISSING] IN A ROW DESIGNATOR.
205	VARIABLE	A ROW DESIGNATOR APPEARS OUTSIDE OF AN ACTUAL PARAMETER LIST OR FILL STATEMENT.
206	VARIABLE	MISSING] .
207	VARIABLE	MISSING [.
208	VARIABLE	WRONG NUMBER OF SUBSCRIPTS.
209	VARIABLE	ARRAYS: PARTIAL WORD DESIGNATOR NOT LEFT-MOST IN LEFT PART LIST.
210	VARIABLE	ARRAYS: MISSING . OR ← .
211	VARIABLE, DBLSTMT	PROCEDURE ID USED OUTSIDE OF SCOPE IN LEFT PART.
212	VARIABLE	SUB-ARRAY DESIGNATOR PERMITTED AS ACTUAL PARAMETER ONLY.
213	MAKEPOINTER	POINTER REQUIRES ARRAY ROW, SUBSCRIPTED VARIABLE, OR ONE-DIMENSIONAL ARRAY ID.
214	STRINGRELATION	POINTER RELATION MUST BE = OR ≠ ONLY.
215	MAKEPOINTER	CHARACTER SIZE MUST BE LITERAL 6 or 8.
216	VARIABLE	LEVEL OR POINTER EXPRESSION EXCEEDS LEVEL OR LEFT-PART POINTER IDENTIFIER.
217	VARIABLE	LEFT-PART POINTER MAY NOT BE CALL-BY-NAME FORMAL PARAMETER.
218	STRINGRELATION	POINTER UPDATE NOT PERMITTED WITH POINTER RELATION.
219	BOOPRIM	RELATIONAL OPERATOR EXPECTED WHEN POINTER UPDATE CONSTRUCT USED.
268	EMITC	A REPEAT INDEX ≥ 64 WAS SPECIFIED OR TOO MANY FORMAL PARAMETERS, LOCALS, AND LABELS.
269	TABLE	A CONSTANT IS SPECIFIED WHICH IS TOO LARGE OR TOO SMALL.
281	DBLSTMT	MISSING (.

APPENDIX J (cont)

COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
282	DBLSTMT	TOO MANY OPERATORS.
283	DBLSTMT	TOO MANY OPERANDS.
284	DBLSTMT	MISSING ,.
285	DBLSTMT	MISSING).
286	DBLSTMT	AN UNDECLARED VARIABLE WAS USED.
300	FILLSTMT	THE IDENTIFIER FOLLOWING THE WORD FILL IS NOT AN ARRAY IDENTIFIER.
301	FILLSTMT, MAKEALABEL	MISSING WITH IN FILL STATEMENT.
302	FILLSTMT	IMPROPER FILL ELEMENT.
303	FILLSTMT	NON OCTAL CHARACTER IN OCTAL FILL. THE THREE LOW ORDER BITS ARE CONVERTED AND COMPILATION CONTINUES.
304	FILLSTMT	IMPROPER ROW DESIGNATOR.
305	FILLSTMT	NUMBER OF DATA WORDS EXCEEDS 1023.
350	CHECKCOMMA	MISSING OR ILLEGAL PARAMETER DELIMITER IN SORT OR MERGE STATEMENT.
351	OUTPROCHECK	ILLEGAL TYPE FOR SORT OR MERGE OUTPUT PROC.
352	OUTPROCHECK	OUTPUT PROCEDURE IN SORT OR MERGE STMT DOES NOT HAVE EXACTLY TWO PARAMETERS.
353	OUTPROCHECK	FIRST PARAMETER OF OUTPUT PROCEDURE MUST BE BOOLEAN.
354	OUTPROCHECK	SECOND PARAM OF OUTPUT PROCEDURE MUST BE ONE-DIM ARRAY.
355	SORTSTMT	MISSING (.
356	HVCHECK	ILLEGAL TYPE FOR SORT OR MERGE HIGHVALUE PRO.
357	HVCHECK	HIVALUE PROCEDURE DOES NOT HAVE EXACTLY ONE PARAMETER.
358	HVCHECK	HIVALUE PROCEDURE PARAM NOT ONE-DIM ARRAY.
359	EQLESCHECK	SORT OR MERGE COMPARE PROCEDURE NOT BOOLEAN.
360	EQLESCHECK	COMPARE PROCEDURE DOES NOT HAVE EXACTLY TWO PARAMETERS.
361	EQLESCHECK	COMPARE PROCEDURE FIRST PARAM NOT 1-D ARRAY.
362	EQLESCHECK	COMPARE PROCEDURE SECOND PARAM NOT 1-D ARRAY.

APPENDIX J (cont)

COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
363	INPROCHECK	SORT STMT INPUT PROCEDURE NOT BOOLEAN.
364	INPROCHECK	INPUT PROCEDURE DOES NOT HAVE EXACTLY ONE PARAMETER.
365	INPROCHECK	INPUT PROCEDURE PARAMETER NOT ONE-D ARRAY.
366	SORTSTMT	MISSING).
367	MERGESTMT	MISSING (.
368	MERGESTMT	MORE THAN 7 or LESS THAN 2 FILES TO MERGE.
369	MERGESTMT	MISSING).
400	MERRIMAC	MISSING FILE ID IN MONITOR DEC.
401	MERRIMAC	MISSING LEFT PARENTHESIS IN MONITOR DEC.
402	MERRIMAC	IMPROPER SUBSCRIPT FOR MONITOR LIST ELEMENT.
403	MERRIMAC	IMPROPER SUBSCRIPT EXPRESSION DELIMITER IN MONITOR LIST ELEMENT.
404	MERRIMAC	IMPROPER NUMBER OF SUBSCRIPTS IN MONITOR LIST ELEMENT.
405	MERRIMAC	LABEL OR SWITCH MONITORED AT IMPROPER LEVEL.
406	MERRIMAC	IMPROPER MONITOR LIST ELEMENT.
407	MERRIMAC	MISSING RIGHT PARENTHESIS IN MONITOR DECLARATION.
408	MERRIMAC	IMPROPER MONITOR DECLARATION DELIMITER.
409	DMUP	MISSING FILE IDENTIFIER IN DUMP DECLARATION.
410	DMUP	MISSING LEFT PARENTHESIS IN DUMP DECLARATION.
411	DMUP	ARRAYS: DUMP LIST HAS WRONG NUMBER OF SUBSCRIPTS OR MISSING RIGHT BRACKET.
412	DMUP	ARRAYS: DUMP LIST HAS WRONG NUMBER OF SUBSCRIPTS OR MISSING COMMA.
413	DMUP	IMPROPER ARRAY DUMP LIST ELEMENT.
414	DMUP	ILLEGAL DUMP LIST ELEMENT.
415	DMUP	MORE THAN 100 LABELS APPEAR AS DUMP LIST ELEMENTS IN ONE DUMP DECLARATION.
416	DMUP	ILLEGAL DUMP LIST ELEMENT DELIMITER.
417	DMUP	MISSING OR NON-LOCAL LABEL IN DUMP DECLARATION.
418	DMUP	MISSING COLON IN DUMP DECLARATION.

APPENDIX J (cont)
 COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
419	DMUP	IMPROPER DUMP DECLARATION DELIMITER.
420	READSTMT	MISSING LEFT PARENTHESIS IN READ STATEMENT.
421	READSTMT	MISSING LEFT PARENTHESIS IN READ REVERSE STATEMENT.
422	READSTMT	MISSING FILE IN READ STATEMENT.
424	READSTMT	IMPROPER FILE DELIMITER IN READ STATEMENT.
425	READSTMT	IMPROPER FORMAT DELIMITER IN READ STATEMENT.
426	READSTMT	IMPROPER DELIMITER FOR SECOND PARAMETER IN READ STATEMENT.
427	READSTMT	IMPROPER ROW DESIGNATOR IN READ STATEMENT.
428	READSTMT	IMPROPER ROW DESIGNATOR DELIMITER IN READ STATEMENT.
429	READSTMT	MISSING ROW DESIGNATOR IN READ STATEMENT.
430	READSTMT	IMPROPER DELIMITER PRECEDING THE LIST IN A READ STATEMENT.
431	FCRSCAN	IMPROPER SYNTAX.
433	HANDLETHETAILEND- OFAREADORSPEACEST- ATEMENT	MISSING RIGHT BRACKET IN READ OR SPACE STATEMENT.
434	SPACESTMT	MISSING LEFT PARENTHESIS IN SPACE STATEMENT.
435	SPACESTMT	IMPROPER FILE IDENTIFIER IN SPACE STATEMENT.
436	SPACESTMT	MISSING COMMA IN SPACE STATEMENT.
437	SPACESTMT	MISSING RIGHT PARENTHESIS IN SPACE STATEMENT.
438	WRITESTMT	MISSING LEFT PARENTHESIS IN A WRITE STATEMENT.
439	WRITESTMT	IMPROPER FILE IDENTIFIER IN A WRITE STATEMENT.
440	WRITESTMT	IMPROPER DELIMITER FOR FIRST PARAMETER IN A WRITE STATEMENT.
441	WRITESTMT	MISSING RIGHT BRACKET IN CARRIAGE CONTROL PART OF A WRITE STATEMENT.
442	WRITESTMT	ILLEGAL CARRIAGE CONTROL DELIMITER IN A WRITE STATEMENT.

APPENDIX J (cont)

COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
443	WRITESTMT	IMPROPER SECOND PARAMETER DELIMITER IN WRITE STATEMENT.
444	WRITESTMT	IMPROPER ROW DESIGNATOR IN A WRITE STATEMENT.
445	WRITESTMT	MISSING RIGHT PARENTHESIS AFTER A ROW DESIGNATOR IN A WRITE STATEMENT.
446	WRITESTMT	IMPROPER DELIMITER PRECEDING A LIST IN A WRITE STATEMENT.
448	WRITESTMT	IMPROPER LIST DELIMITER IN A WRITE STATEMENT.
449	READSTMT	IMPROPER LIST DELIMITER IN A READ STATEMENT.
450	LOCKSTMT	MISSING LEFT PARENTHESIS IN A LOCK STATEMENT.
451	LOCKSTMT	IMPROPER FILE PART IN A LOCK STATEMENT.
452	LOCKSTMT	MISSING COMMA IN A LOCK STATEMENT.
453	LOCKSTMT	IMPROPER UNIT DISPOSITION PART IN A LOCK STATEMENT.
454	LOCKSTMT	MISSING RIGHT PARENTHESIS IN A CLOSE STATEMENT.
455	CLOSESTMT	MISSING LEFT PARENTHESIS IN A CLOSE STATEMENT.
456	CLOSESTMT	IMPROPER FILE PART IN A CLOSE STATEMENT.
457	CLOSESTMT	MISSING COMMA IN A CLOSE STATEMENT.
458	CLOSESTMT	IMPROPER UNIT DISPOSITION PART IN A CLOSE STATEMENT.
459	CLOSESTMT	MISSING RIGHT PARENTHESIS IN A CLOSE STATEMENT.
460	RWNDSTMT	MISSING LEFT PARENTHESIS IN A REWIND STATEMENT.
461	RWNDSTMT	IMPROPER FILE PART IN A REWIND STATEMENT.
462	RWNDSTMT	MISSING RIGHT PARENTHESIS IN A REWIND STATEMENT.
463	BLOCK	A MONITOR DECLARATION APPEARS IN THE SPECIFICATION PART OF A PROCEDURE.
464	BLOCK	A DUMP DECLARATION APPEARS IN THE SPECIFICATION PART OF A PROCEDURE.
465	DMUP	DUMP INDICATOR MUST BE UNSIGNED INTEGER OR SIMPLE VARIABLE.
500	SEARCHLIB	ILLEGAL LIBRARY IDENTIFIER.

APPENDIX J (cont)
COMPATIBLE ALGOL COMPILER ERROR MESSAGES

<u>ERROR NUMBER</u>	<u>ROUTINE</u>	<u>ERROR MESSAGE</u>
501	SEARCHLIB	LIBRARY IDENTIFIER NOT CONTAINED IN DIRECTORY.
502	SEARCHLIB	ILLEGAL LIBRARY START POINT.
503	SEARCHLIB	SEPARATOR REQUIRED BETWEEN START POINT AND LENGTH.
504	SEARCHLIB	ILLEGAL LIBRARY LENGTH.
505	SEARCHLIB	MISSING BRACKET.
507	SEARCHLIB	TAPE POSITIONING HARDWARE-FAILURE.
508	Anywhere	CONSTRUCT NOT ALLOWED IN TIME SHARING SYSTEM.
509	IODEC	NON-LITERAL FILE VALUE NOT GLOBAL TO FILE DECL.

APPENDIX K
THE BREAKOUT PROCESS

ALGOL programs create a permanent break file on disk when a break statement is executed. COBOL programs act similarly when a RERUN Statement is executed or a RERUN Clause is invoked, with one exception: A COBOL Reel RERUN Clause specifies that the break file is placed on the beginning of an output reel of a tape file.

In this instance, a temporary break file is created on disk and is copied onto the tape; the disk file is destroyed on completion or termination of the copy.

The ONEBREAK Option is ignored.

Tape, disk, and pseudo reader files, and line print files, regardless of back-up label equations, are permitted to be open when a break is done. Sort, card reader, and data communications files preclude breaking. If a break is attempted while one of the forbidden files is open, the break attempt is ignored; and the operator is told which file interfered with the break.

Breaks are numbered cyclically from 00 through 99. The numbering of break files following a restart is the same as would occur if a restart was not done. Thus, if break file number 06 is used to restart a job, the next break file created by the file is 07.

Provision has been made for the handling of write errors when the reel option is evoked, forcing the back-up file onto tape. If an error occurs, the operator is notified of a BADUMP, the reel is switched, and a new break file, having the same break number, is built.

Break files are program files named according to their break number and to whichever program built them. Thus, break files, created when executing the program BREAKER/A, are named BREAKER/BREAKnn, where each nn is a break number. The files have 30 words per record, and the records are allocated in 500-record chunks.

APPENDIX K (cont)
THE BREAKOUT PROCESS

Break files differ from other program files, although segment zero is somewhat similar. A comment, detailing the differences of segment zero, follows the DCMCP Procedure BREAKOUT.

A copy of the currently coded overlay disk of the job and intrinsics which the job might use are included in the break file. The file also has a map for exact replacement of some of the core areas of the job, primarily the nonoverlayable areas.

Breakout usually takes about five seconds building a 2-chunk break file, during which time the system is disk bound. It uses about 1.5 K of core storage.

Breaking does not affect the contents of files; however, it is necessary to save temporary files. Both temporary and permanent disk files are entered into the Directory when a break occurs. Tape files are marked to be saved.

All label equation information is retained and is used to recognize files at restart.

RESTARTING PROCESS.

If the reel RERUN Clause is specified, the break file must be loaded from tape onto disk. This is accomplished by the RS<unit> message which checks to determine if the mentioned unit is a labeled, write-enabled tape with a break file copy. If so, the system loads the copy unless there is already a permanent disk file with the same name as that of the copy. After loading, the unit is left marked and positioned for the pending restart. If the load try fails, the unit is set not-in-use and locked. Note that the RS Message only causes re-loading of the break files; it does not initiate restarts.

Files reopened while restarting must correspond closely to those open when the break file is built; the reopening files are therefore

APPENDIX K (cont)
THE BREAKOUT PROCESS

thoroughly checked before acceptance. The type of a file may not be changed between break and reopening. File reopening is detailed below.

To restart, the appropriate break file is executed or run. If the MCP is not compiled with the \$ SET BREAKOUT=TRUE Module, the break file is considered to be nonexecutable code.

Mapped areas must be replaced exactly where they are at breakout. This replacement is considered to be the "restarting" process. When replacement is complete, the operator is notified that the job has restarted; and the system begins reopening the files of the job. Replacement side-effects are presented below. Restarts must wait for particular areas to become available; therefore, restarts should be performed without jobs in the mix, preferably immediately after a Halt/Load.

All control cards used in originally initiating the program are used to restart that program. The stack, label equation, and common cards are ignored by the restarting process.

Once a restart job has been initiated (BOJ) and until the restarting process is complete, the system is occupied by replacing or rebuilding the core areas of the job. It is possible that the entire address range required is not available, e.g., that some in-use area is interfering with replacement. Simply waiting resolves some interference. Some areas must be moved or denied space during restarts since it is not possible to wait until the interference clears up.

There are various consequences of replacement. The restart is automatically ESed if the memory configuration differs from that at break. The IN, OT, and ST Operator Requests and changing intrinsics (either with a CI or XI) are not valid when applied to restarting jobs.

APPENDIX K (cont)
THE BREAKOUT PROCESS

The operator may monitor the progress of a job in restarting. If the operator requests <mix>WY, he is advised if the restart is waiting because of interfering areas. Rarely, however, should the MCP (mix zero) or the restarting job cause interference.

A restart job that does not have an excessive amount of overlay disk usually takes about four seconds restarting; none of its storage is overlayable. A restarting job uses less core storage than it does breaking. However, when resolving interferences involves moving many areas, a NO MEM situation may occur. In this event, movement is temporarily abandoned, and the system recovers; a Halt/Load and a new restart attempt may be required.

FILE HANDLING.

After a job has restarted, its files are "reopened." The system finds files, checks them against data saved at break, and then repositions them forward accordingly. If a file checked is somehow unsuitable, the operator is advised of what is wrong; and the system tries finding the correct file.

Only tape files are spaced. The restart job is terminated if an irrecoverable parity or other problem occurs. Positioning other files is ignored; their proper positioning derives from internal references left intact.

It should be noted that file changes made after a break are not considered while reopening the file. For instance, errors may result if a record is added, deleted, or altered. Such errors need not appear immediately after the breakpoint.

The MT, DK, CD, LP, PBT, and PBD Files of a job may be open at break. Reopening these files is discussed below.

The checks which apply should be noted carefully. To break at all, other types of files and sort files must be closed; they are ignored while the system reopens files.

APPENDIX K (cont)
THE BREAKOUT PROCESS

LP, PBT, and PBD Files need no attention while breaking. At restart, line-print continues with whichever line logically follows the break. The line printer selected is neither positioned within the page nor physically labeled. Back-up files restart similarly but with new files. Only type correspondence is checked.

Disk files are permanent or nonpermanent according to whether they are mentioned in the Directory at break; temporary files become permanent. Their save factor is set to 63 if previously zero or unspecified. Information in the Directory about permanent files in use may be inaccurate; therefore, it is updated during each break.

At restart, disk files are checked for blocking, record size, number and use of rows, allowed number of rows, and end-of-file suitability.

Pseudo readers are checked for type correspondence only. Normal checks in the system detect short control-decks, and the other disk file checks are not relevant. However, control decks have internal control card linkages, references to which reopening does not check. Furthermore, the system cannot find a subdeck unless it is current in a pseudo reader.

Tape files (not PBT) have a physical block count recorded during each break. Finding an input tape file may involve searching a multifile. Reopening tape files are found as though they are input files; once checked and accepted they are spaced forward according to the difference between current and break counts.

Thus, a tape is effectively positioned relative to the beginning of its reel.

As accepted, tapes are positioned in parallel.

If an output tape reel is RSed to load the break just restarted, the tape is checked by the RS Handler; its unit is left not-in-use, positioned, and marked to prevent nonrestart jobs from finding it. The

APPENDIX K (cont)
THE BREAKOUT PROCESS

unit is therefore checked for marks left by the RS Handler; if the unit is unmarked, a tape is found, checked like other tapes, and also checked for a break file copy.

Other tapes (not PBT) are checked for type, label, "format," dump, and write-enable correspondence; tape "format" correspondence is wrong if the requested file is found after the block is broke. Tape label correspondence is wrong if the broken file has a label and the file found has no label. Dump correspondence is wrong if the file found lacks a needed break file copy. Write-enable correspondence is wrong if the file needs (lacks) a write ring.

SYSTEMS EFFECTS.

Breakout-restart has been redesigned to eliminate its characteristic maintenance requirement. With this end in view, the new design confines its attention to the job principally involved, thereby markedly reducing tangential interactions with the system and other jobs. Two of such remaining interactions warrant attention.

It should be noted that breakout saves all intrinsics the job may use. This ensures the integrity of references thereto, but users may occasionally restart with functionally dated intrinsics, resulting in an error. To avoid potential problems, the same MCP and intrinsics used at breakout time should be used at restart time.

It should be noted that restart storage replacement involves moving areas. Such a movement must only occur when all extant references to the area can be corrected. Therefore, any MCP changes must be made cooperative with breakout/restart requirements. If control is lost with a new reference to such an area pending or if new areas are introduced, it should be considered whether and how the areas should be moved.

BREAKOUT MESSAGES.

--CAN-T BREAK <data file designator> <rdc>: <job specifier>

APPENDIX K (cont)
THE BREAKOUT PROCESS

The job tried breaking with a file of unsuitable type of open. The break try is ignored.

<priority>: <job specifier>=<mix>: BREAK<break number> BUILT
The specified job just broke, creating the break file <program name>/BREAK<break number>. The break file is then moved to an output tape if necessary.

<priority>: <job specifier>=<mix>: BADUMP ON <unit>
The system could not copy a break file onto the mentioned tape. It will try again on a new reel.

THE RS MESSAGE.

This message allows the operator to add a break file to the Directory, the break file having been copied to the output tape of a COBOL job. The RS message format is:

RS<unit>

The responses are:

- a. RS<unit> INV KBD
The unit is not a tape.
- b. <unit> <note>
The unit must be an available, labeled, write-enabled tape with a break file copy. The note is either NOT READY, IN USE, SCRATCH, WRITE LOCK, or NO DUMP.
- c. .<program name>/BREAK<break number> NOT ADDED.DUP LIB
RS<unit>
There is already a disk file with the names mentioned. The break file on the unit is not loaded.
- d. <unit> ERROR IN DUMP
The tape-disk break file copy went awry. The break file is not loaded.

APPENDIX K (cont)
THE BREAKOUT PROCESS

- e. <program name>/BREAK<number> ADDED.TAPE POSITIONED:<unit>
The RS<unit> is successful. The unit is left positioned and marked for the pending restart of the loaded break file.

RESTART MESSAGES.

<job specifier>=<mix index> GONE <time>

The job is a restart and was ESed or DSed before having restarted.

<priority>:<job specifier>=<mix index> RESTARTED

The designated restart job has completed replacing core storage and now has a normal job structure; i.e., it has "restarted." The job will begin reopening files next.

<priority>:<job specifier>=<mix> RESTART IS <state>

The <state> is either WAITING or MOVING. The operator requested <mix>WY before the job restarted. This response tells what restart is doing.

--MIX: <mix>, ..., <mix> IN THE WAY

The mixes, e.g., zero for the MCP, are using core areas needed to replace storage of the restarting job.

FILE REOPENING MESSAGES.

--WRONG FILE <data file designator> <rdc>:<job specifier>

Reopening involves checking files for compatibility with those in use at breakout. The designated file is not sufficiently compatible; the next message hints why. The operator may respond with an OK, WY, or DS reply; OK initiates a recheck.

--WRONG <hint>, ..., <hint>

The hints are among the following:

- a. WRITE STATE

The write ring of the file is in the wrong place (in the box or on the tape).

APPENDIX K (cont)
THE BREAKOUT PROCESS

- b. LABEL
The file lacks (needs) a label.
- c. TYPE
A file on disk (tape, line printer) at break must be on disk (tape, line printer) at restart. For example, an LP File was broken, and the operator tried reopening it as a PBD.
- d. ROWS USED
Disk files have up to 20 rows. The one found does not have the right one.
- e. NO. OF ROWS
The disk file found has the wrong number of rows allowed.
- f. EOF
It is too short.
- g. ROW LENGTH
Its rows are the wrong size.
- h. FORMAT
If it is a disk file, its blocking or record length is wrong. If it is a tape file, it was found beyond where it was in breakout.
- i. SECURITY
A security error occurred.
- j. NO DUMP
The tape file lacks a break file copy.



APPENDIX E
CONSTRUCTION OF COLD START
AND COOL START DECKS

It is possible to COOL START from the MARK X to the MARK XI system rather than doing a COLD START. The only condition necessary for this capability is that DIRECTORYTOP in both systems must be equal. The MARK XI COOL START is used in going from MARK X to MARK XI and on the MARK XI system, and the MARK X COOL START is used in going from MARK XI to the MARK X system.

If a transition is made from MARK XI to MARK X, a Halt/Load should be done on the MARK XI system immediately before the MARK X COOL START is run to zero the open counts in the disk file header.

In the following sample deck setups, the "KERNEL" has been included in the COLD START and COOL START decks under the belief that if something has happened to disk so that a Halt/Load is impossible, a COOL START should be done. If the COOL START followed by a Tape to Disk is not sufficient to bring the system up, a COLD START is necessary.

The Tape to Disk loader, Disk to Disk loader, and "KERNEL" may also be used independently of the COLD/START and COOL/START programs. Each object deck must be preceded by a one card ESPOL loader. If the "KERNEL" is used independently, it will only initiate a Halt/Load, the "KERNEL" will not be placed on disk.

COLD START DECK

1. One card ESPOL loader.
2. COLD START object program.
3. "KERNEL" object program.
4. COLD START parameter deck.
 - A. DATE card.
 - B. DIRCTRYTP card.



- C. DIRECT card.
- D. ESU card.
- E. SYSTEM card.
- F. FENCE card.
- G. FILE cards.

NOTE

A FILE card must be present in the COLD START deck for the MCP to be loaded from tape with the Tape to Disk loader.

I.e. FILE MCP/DISK, 1x1170, 999

- H. Option cards.
 - I. STOP card.
5. One card ESPOL loader.
 6. Tape to Disk loader object program.
 7. OPTIONAL Tape to Disk loader parameter cards.
 - A. TAPE card.
 - B. FILE card.

COOL START DECK

1. One card ESPOL loader.
2. COOL START object program.
3. "KERNEL" object program.
4. COOL START parameter deck.
 - A. DATE card.
 - B. DIRCTRYTP card.
 - C. DIRECT card.
 - D. ESU card.
 - E. SYSTEM card.
 - F. FENCE card.
 - G. Option cards.
 - H. STOP card.

ESPOL LOADER
TAPE TO DISK
PARAM CARD

"MCP/DISK"

INDEX

- ALGOL \$\$ card, 4-41
- ALGOL error messages, F-1
- ALGOL source program, 4-43
- auxiliary memory, 2-8
- auxiliary stacker full,
card punch, 2-58

- basic change deck, 3-6
- breakout process, K-1
- buffer conditions (DTTU), 2-141

- card jam, card reader, 2-13,
2-28, 2-34A
- Card Load Select Programs, 3-14
- card not at prepunch station,
card punch, 2-57
- card not at read station, card
punch, 2-43
- card not at ready station, card
punch, 2-43
- card punches, 2-34A, 2-45, 2-59
- card readers, 2-8B, 2-17, 2-33
- card reader control deck
file, 5-2
- carriage control tape insertion,
line printers, 2-74, 2-78J
- chad receptacle, paper tape
punch, 2-106
- CHANGE card, 4-16A
- channel select plugboard,
paper tape reader, 2-80
- channel select plugboard,
paper tape punch, 2-95
- COBOL \$\$ card, 4-42
- COBOL Compiler error and
diagnostic messages, G-1
- COBOL program, 4-29
- COBOL source program, 4-43

- code translator, paper tape
punch, 2-95
- code word, 5-20
- Cold Start deck, 3-19
- Cold Start Program, 3-15
- COMMON card, 4-24
- Compatible ALGOL Compiler
error messages, J-1
- Compile-and-Go run, 4-9
- COMPILE card, 4-9
- Compile-for-Library run, 4-9
- Compile-for-Syntax-Check run, 4-10
- compiler and object program
information, 5-22
- compiler option cards, 4-26
- compiling source, 3-7
- control card errors in pseudo
card decks, 5-5
- control card information, 5-20
- control card syntax, 5-6
- control cards, 4-8A
- control cards for system
loading, 3-36
- control cards used to load
compilers onto disk, 3-37
- control deck onto tape,
copying a, 5-3
- control information, 4-1
- control information via
punched cards, 4-8
- control panel,
 - B 122, 2-9
 - B 123, 2-19
 - B 5005, 2-8A
 - B 5350, 2-145
 - B 9111, 2-34

INDEX (cont)

control panel (cont)

B 9120, 2-84
B 9210, 2-34D
B 9211, 2-47
B 9213-1, 2-61
B 9220, 2-100
B 9240, 2-62H
B 9242-4, 2-78B
B 9373, 2-132
B 9396, 2-110
B 9410, 2-124

console, 2-2

supervisory printer, 2-5

Cool Start deck, 3-35

Cool Start Program, 3-16

CORE card, 4-25

Core to Tape Dump Program, 3-17

cover not in place,

B 122, 2-15

B 123, 2-29

B 9211, 2-58

cover opened,

B 9210, 2-43

B 9213-1, 2-62F

DATA card, 4-16B

data communications
processor, 2-143

data communications
system, 1-3, 2-138

data transmission control
unit, 2-139

data transmission terminal
unit, 2-140

DATE card, 3-22

deck structure, 4-49

DIRECT card, 3-20

Disk Directory, 5-41

disk failure, C-5

disk file, 2-128

disk file control unit, 2-130

disk file/data communications
cabinet, 2-129

disk file/data transmission
terminal unit cabinet, 2-129

disk file electronics unit, 2-132

disk file expanded control, 2-129

disk file system, 2-128

disk Halt/Load card, 3-37

disk lockout switches, 2-134

Disk to Disk MCP Loader deck, 3-18

Disk to Disk MCP Loader
Program, 3-17

Dollar Sign (\$) card, 4-27

DRCTRYTP card, 3-20

DUMP card, 4-12

END card, 4-17

END CONTROL card, 4-17

end-of-job and error
messages, 5-18

END OF PAPER indicator lit,
line printers, 2-76, 2-78L

error stacker full,
card punch, 2-58

ESPOL, B-2

ESPOL Loader Program, 3-15

ESU card, 3-21

EXECUTE card, 4-10

FEED CHECK indicator lit,

B 122, 2-15

INDEX (cont)

- FEED CHECK indicator lit (cont)
 - B 123, 2-30
 - B 9210, 2-38
 - B 9211, 2-53
- feed error condition,
 - card reader, 2-34A
- feed roll block not locked,
 - card punch, 2-56, 2-62E
- FENCE card, 3-22
- FILE card (label equation), 4-22
- FILE card group, 3-23
- file descriptions, 5-35
- file opening action, 5-47
- file protect memory, 2-136
- file security system, E-1
- format of blocks on a PB file, 5-45
- format of printer backup file on disk, 5-46
- format of records on PB files, 5-45
- forms handling, line printer, 2-63, 2-78C
- forms, special, 5-47
- FORTTRAN Compiler, B-1
- FORTTRAN Compiler error messages, H-1
- FORTTRAN Translator control cards, 4-43
- FORTTRAN Translator error messages, I-1
- Halt/Load Kernel Program, 3-17
- Halt/Load Program, 3-15
- hopper empty,
 - B 122, 2-15
 - B 123, 2-29
- hopper empty (cont)
 - B 9210, 2-43
 - B 9211, 2-56
 - B 9213-1, 2-62C
- input code translator,
 - paper tape, 2-81
- IO card, 4-19
- LABEL card, 4-16A
- LDCNTRL/DISK Program, 5-1
- leaders, attaching magnetic tape, 2-118
- line printers, 2-62G, 2-78
- line selection knob in N position, line printer, 2-77
- LOAD card - ADD card, 4-15
- loading a control deck file onto disk, 5-1
- loading and maintaining the system, 3-1
- loading paper tape, 2-102
- loading the system from the SYSTEM tape, 3-38
- loading the magnetic tape supply reel, 2-112
- loading the magnetic tape take-up reel, 2-117
- log entry specifications, 5-19
- log initializing, 5-26A
- log maintenance, 5-19
- logging of PB files, 5-48
- magnetic tape,
 - care, 2-122
 - control deck file, 5-2
 - handling, 2-123

INDEX (cont)

- magnetic tape (cont)
 - library procedures, 2-124
 - loading, 2-123
 - splicing, 2-120
 - storage, 2-122
 - units, 2-108
- maintenance function
 - examples, 5-10
- MCP Loader decks, 3-17
- MCP modularity, 4-33
- messages, C-1
- Multiprocessing Factor, 3-41
- Nines card, 4-43
- Not Ready conditions,
 - B 122, 2-12
 - B 123, 2-26
 - B 9111, 2-34A
 - B 9210, 2-37
 - B 9211, 2-52
 - B 9213-1, 2-62B
 - B 9240, 2-76
 - B 9242-4, 2-78K
- operating procedures,
 - B 122, 2-11
 - B 123, 2-21
 - B 9111, 2-34
 - B 9120, 2-87
 - B 9210, 2-36
 - B 9211, 2-49
 - B 9213-1, 2-62A
 - B 9242-4, 2-78D
- operator console, 2-1
- operator maintenance,
 - B 122, 2-16
 - B 123, 2-32
 - B 9111, 2-34A
 - B 9120, 2-92
 - B 9210, 2-44
 - B 9211, 2-58
 - B 9213-1, 2-62F
 - B 9220, 2-107
 - B 9240, 2-77
 - B 9242-4, 2-78L
 - B 9396, 2-121
- OPTION cards, 3-26
- OPTN 18 card, 3-35
- OPTN 19 card, 3-34
- OPTN 20 card, 3-34
- OPTN 21 card, 3-34
- OPTN 22 card, 3-34
- OPTN 23 card, 3-34
- OPTN 24 card, 3-34
- OPTN 25 card, 3-33
- OPTN 26 card, 3-33
- OPTN 27 card, 3-32
- OPTN 28 card, 3-32
- OPTN 29 card, 3-32
- OPTN 30 card, 3-31
- OPTN 31 card, 3-31
- OPTN 32 card, 3-31
- OPTN 33 card, 3-31
- OPTN 34 card, 3-30
- OPTN 35 card, 3-30
- OPTN 36 card, 3-30
- OPTN 37 card, 3-29

INDEX (cont)

- OPTN 38 card, 3-29
- OPTN 39 card, 3-29
- OPTN 40 card, 3-28
- OPTN 41 card, 3-28
- OPTN 42 card, 3-28
- OPTN 43 card, 3-27
- OPTN 44 card, 3-27
- OPTN 45 card, 3-27
- OPTN 46 card, 3-26
- OPTN 47 card, 3-26

- paper slewing, line printer, 2-77, 2-78L
- paper tape punch, 2-93
- paper tape reader, 2-78M
- paper tape splicing, 2-106
- parity on a control deck magnetic tape file, 5-4
- patches, merging, 3-6
- peripheral switching unit, 2-124
- primary stacker full (B 9211), 2-58
- print drum not in position, line printer, 2-77, 2-78L
- print file on disk, closing a, 5-48
- printer backup information, 5-44
- PRIORITY card, 4-21
- PROCESS card, 4-19
- program-parameter cards, 4-18
- program scheduling information, 3-39
- pseudo card readers and the use of pseudo decks on disk, 5-4
- pseudo decks on disk, 5-2
- punch block not locked (B 9211), 2-56
- PUNCH CHECK indicator lit (B 9210), 2-43
- PUNCH CHECK indicator lit (B 9211), 2-58
- PUNCH CHK indicator on (B 9213-1), 2-62F
- punch die not in place (B 9210), 2-43
- punch mechanism not locked (B 9213-1), 2-62E

- Read Check condition (B 123), 2-27
- READ CHECK indicator lit (B 122), 2-15
- READ CHECK indicator lit (B 123), 2-30
- record n + 1, 5-26B
- record size error, G-23
- record zero, 5-26A
- records, special, 5-26A
- REMOVE card, 4-11
- RESET ACCESSSD card, 4-16B
- ribbon changing, line printer, 2-69, 2-78G
- RN message to turn off pseudo card readers, 5-5
- RN message to turn on pseudo card readers, 5-4

- SAVE card, 4-26
- scheduling from disk, 5-1
- Selection Algorithm, 3-40
- SET ACCESSSD card, 4-16B
- shared disk system, 2-136
- software, 3-1
- source program cards, 4-43
- STACK card, 4-20

INDEX (cont)

- stacker full,
 - B 122, 2-14
 - B 123, 2-29
 - B 9210, 2-44
 - B 9211, 2-58
 - B 9213-1, 2-62C
- standard system log, 5-34
- statistics log, 5-32
- statistics log file, 5-38
- STOP card, 3-35
- STOP switch pressed,
 - B 122, 2-15
 - B 123, 2-30
 - B 9210, 2-43
 - B 9211, 2-56
 - B 9213-1, 2-62D
 - B 9240, 2-77
 - B 9242-4, 2-78L
- supervisory printer, 2-4A
- SYMBOL tape, 3-1
- symbolic library file on disk, 5-5
- symbolic library tapes onto disk, copying, 5-19
- system design, 1-2
- System Loader decks, 3-17
- system procedures, 3-37
- system start-up procedure, 3-38
- system statistics file, 5-35
- SYSTEM tape, 3-14
- SYSTEMS card, 3-21
- systems material, 3-1
- systems memory storage module, 2-134
- tape and forms registration, line printer, 2-75, 2-78K
- tape movement, stopping paper tape reader, 2-90
- tape punching, line printer carriage control, 2-64, 2-78C
- tape rewinding, magnetic, 2-118
- tape rewinding, paper, 2-105
- Tape to Disk MCP Loader deck, 3-18
- Tape to Disk MCP Loader Program, 3-16
- time sharing log, 5-34
- time sharing log additions, 5-39
- TYPE BOJ card, 3-27
- TYPE CLOSE card, 3-30
- TYPE CMLFILE card, 3-30
- TYPE DATE card, 3-28
- TYPE DISCONDC card, 3-30
- TYPE DISKLOG card, 3-34
- TYPE DISKMSG card, 3-34
- TYPE EOJ card, 3-27
- TYPE ERRORMSG card, 3-31
- TYPE LIBERR card, 3-34
- TYPE LIBMSG card, 3-31
- TYPE OPEN card, 3-27
- TYPE PBDREL card, 3-33
- TYPE RET card, 3-31
- TYPE RSMMSG card, 3-34
- TYPE SCHEDMSG card, 3-31
- TYPE SECMSG card, 3-32
- TYPE TIME card, 3-28
- UNIT card, 4-25
- unloading cards from the card punch, 2-37, 2-51, 2-62B

INDEX (cont)

unloading paper tape, 2-91, 2-105

unloading the magnetic tape supply
reel, 2-116

unloading the magnetic tape
take-up reel, 2-118

USE AUTOUNLD card, 3-35

USE AUTOPRNT card, 3-29

USE CHECK card, 3-33

USE CLEARWRS card, 3-29

USE DRA card, 3-26

USE DRB card, 3-26

USE DSKTOG card, 3-32

USE PBDONLY card, 3-34

USE RELTOG card, 3-32

USE SAVEPBT card, 3-34

USE TERMINATE card, 3-28

utility routines, 5-1