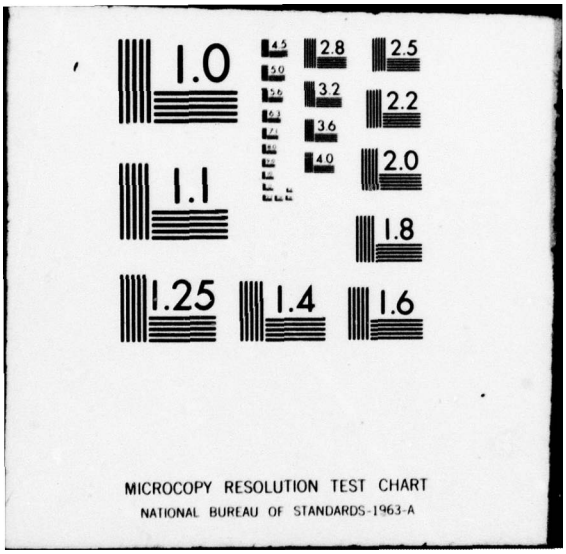
AD-A063 431 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 9/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78 C K HILL
UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-6 SBIE-AD-E100 131 NL

1 OF 4
AD
AD A063431


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-E100 131

COMPUTER SYSTEM MANUAL
CSM UM 15-78
VOLUME VI
1 SEPTEMBER 1978

LEVEL III

COMMAND & CONTROL TECHNICAL CENTER

NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

VOLUME VI
TERMINAL PROCESSING (TP)

USERS MANUAL

DEFENSE COMMUNICATIONS AGENCY

DDC FILE COPY
AD A063431

DDC
RECEIVED
JAN 18 1979
B

THIS DOCUMENT HAS BEEN APPROVED FOR PUBLIC RELEASE AND SALE; ITS DISTRIBUTION IS UNLIMITED.

78 12 06 017

(9) Computer system manual, (12) 350p.

COMMAND AND CONTROL TECHNICAL CENTER

Computer System Manual Number CSM UM 15-78

(11) 1 September 1978

(6)

NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (NIPS 360 FFS)

User's Manual

Volume VI, Terminal Processing (TP)

(14) CCTC-CSM-UM-15-78-VOL-6

(18) SBIE

SUBMITTED BY:

(19) AD-E100 131

APPROVED BY:

(10) CRAIG K. HILL
~~Captain, USA~~
CCTC Project Officer

FREDERIC A. GRAF, JR.
Captain, U.S. Navy
Deputy Director
NMCS ADP

Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314

This document has been approved for public release and sale; its distribution is unlimited.

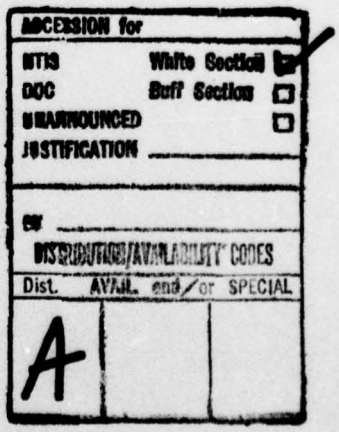
DDC
RECEIVED
JAN 18 1979
B

409 658

78 12 06 017

ACKNOWLEDGMENT

This manual was prepared under the direction of the Chief for Programming with general technical support provided by the International Business Machines Corporation under contracts DCA 100-67-C-0062, DCA 100-69-C-0029, DCA 100-70-C-0031, DCA-70-C-0080 DCA 100-71-C-0047 and DCA 100-77-C-0065.



CONTENTS

Section		Page
	ACKNOWLEDGMENT.....	ii
	ABSTRACT.....	xiii
1	INTRODUCTION.....	1
2	TERMINAL PROCESSING (TP).....	4
2.1	Supported Terminal Devices.....	5
2.1.1	IBM 2260 Display Station Operation..	6
2.1.1.1	Display Screen.....	6
2.1.1.2	Entering Message.....	6
2.1.1.3	Printing the Contents of the Scope..	7
2.1.2	IBM 2250 Display Unit Operation.....	11
2.1.2.1	Entering Message.....	11
2.1.2.2	Cursor Positioning.....	11
2.1.3	IBM 1050 Dial-Up Terminal System Operation.....	11
2.1.3.1	Terminal Switches.....	12
2.1.3.2	1050 Sign-On Procedure.....	12
2.1.3.3	Entering Messages.....	13
2.1.3.4	Terminal Usage.....	13
2.1.3.5	Sign-Off.....	13
2.1.4	IBM 2741 Nonswitched Terminal Operation.....	14
2.1.4.1	Transmission Controls.....	14
2.1.4.2	Transmitting and Receiving With a 2741.....	14
2.1.4.3	2741 Sign-on Procedure.....	15
2.1.4.4	Entering Messages.....	15
2.1.4.5	2741 Terminal Usage.....	16
2.1.4.6	Sign-Off.....	16
2.1.5	IBM 3270 Information Display System Operation.....	16
2.1.5.1	Special Features.....	17
2.1.5.2	Function of 3277 Typewriter Keyboard.....	19
2.1.5.3	Entering a Message.....	21
2.1.5.4	PAGE Display Commands for Formatted 3277.....	21
2.1.6	General CRT Terminal Usage.....	22
2.2	TP LOGON/LOGOFF Procedure.....	23
2.2.1	LOGON Request.....	24
2.2.1.1	LOGON Requirements for 3270 Terminal	26
2.2.2	LOGOFF Request.....	26
2.2.3	REMARKS Request.....	27
2.3	Input Messages.....	27
2.3.1	TP Application Program Input.....	28

Section		Page
2.3.2	Conversational Input.....	29
2.3.3	TP Control Messages.....	29
2.3.3.1	Program Request Message (Logical NOT) Program Name.....	29
2.3.3.2	Abort Message (Logical NOT) STOP.....	30
2.3.3.3	Correction Message (Logical NOT) nn..	30
2.3.3.4	Time Request (Logical NOT) TIME.....	31
2.3.4	Input Message Summary.....	32
2.3.5	Responses to Input Messages.....	32
2.3.5.1	Normal Progress Messages.....	33
2.3.5.2	Warnings and Errors.....	33
2.3.5.2.1	While Paging Through Output.....	33
2.3.5.2.2	In Response to a ¬X Program Request.	35
2.3.5.2.3	In Response to a Line of Input Other Than Paging Commands.....	37
2.4	Output.....	38
2.4.1	Output Display.....	38
2.4.1.1	Page Display Format (2260 Display Station).....	39
2.4.1.2	Display Commands (2260 Display Station).....	41
2.4.1.3	Display Examples (2260 Display Station).....	44
2.5	Utility Functions.....	51
2.5.1	List Input Message Queue (LIST).....	51
2.5.2	Terminal/Operator Communication.....	54
2.5.2.1	Message to Operator (MTO).....	54
2.5.2.2	Operator to Terminal.....	56
2.5.3	TPDUMP Program.....	58
2.5.3.1	TPDUMP Use.....	58
2.5.3.2	Simultaneous Operation from Multiple Terminals.....	62
2.5.3.3	DUMP Examples.....	63
2.5.3.4	DUMP Errors.....	64
2.5.4	Terminal/Terminal Communication.....	64
2.5.4.1	BLAST.....	65
2.5.4.1.1	BLAST Errors.....	67
2.5.4.2	ACCESS.....	68
2.5.5	UTILITY Program.....	68
2.6	Independent Graphic Support (IGS)...	70
2.6.1	Terminal Operations Using IGS.....	70
2.7	VIEW.....	70
2.7.1	Creating A Distribution Data Set....	71
2.7.2	Using VIEW With A 3270 Terminal.....	74
2.7.3	Sample 3270 Screen Produced By VIEW.	74
2.7.4	Using VIEW with 2260 Terminal.....	75
2.7.5	Sample 2260 Screen Produced by VIEW.	76
3	QUICK INQUIRY PROCESSOR (QUIP).....	77
3.1	QUIP Execution.....	77

Section		Page
3.2	QUIP Operating Environment.....	79
3.2.1	File Analysis and Run Optimization Statistics.....	79
3.2.2	Accessing a SAM Data File From a Terminal.....	81
3.2.3	Subfile Function.....	82
3.2.4	Non-NIPS File Processing.....	83
3.2.5	PBSIZE Parameters.....	84
3.2.6	PBSIZE1 Parameter.....	84
3.3	QUIP Source Language Formatting.....	85
3.4	Output Page Format.....	86
3.5	Operators Available in QUIP.....	88
3.5.1	Run Initializing Operators.....	89
3.5.1.1	FILE(QUERY) Operator.....	90
3.5.1.2	CLASS Operator.....	93
3.5.2	Page Environment Operators.....	95
3.5.2.1	HEADER Operator.....	95
3.5.2.2	TRAILER Operator.....	96
3.5.2.3	PAGENO Operator.....	98
3.5.2.4	SYSDATE Operator.....	98
3.5.2.5	NLINES Operator.....	99
3.5.3	Imperative Operators.....	100
3.5.3.1	LIST Operator.....	101
3.5.3.2	DISPLAY Operator.....	105
3.5.3.3	MARK Operator.....	109
3.5.3.4	PRINT Operator.....	110
3.5.4	Arithmetic Operators.....	111
3.5.4.1	SUM Operator.....	111
3.5.4.2	COUNT Operator.....	112
3.5.4.3	COMPUTE Operator.....	113
3.5.4.3.1	Arithmetic Operations.....	116
3.5.4.3.2	Statistical Operations.....	117
3.5.4.3.2.1	AVERAGE Function.....	117
3.5.4.3.2.2	PERCENT Function.....	117
3.5.4.3.2.3	VARIANCE Function.....	118
3.5.4.3.2.4	MAXIMUM Function.....	119
3.5.4.3.2.5	MINIMUM Function.....	119
3.5.4.3.2.6	Examples of Statistical Operators..	119
3.5.5	Conditional Operators.....	121
3.5.5.1	Output Type IF Operator.....	121
3.5.5.1.1	INITIAL Modifier.....	123
3.5.5.2	FINAL Operator.....	124
3.5.6	Forms Control Operators.....	124
3.5.6.1	SPACE Operator.....	124
3.5.6.2	EJECT Operator.....	125
3.5.7	Matrix Generation Operators.....	125
3.5.7.1	FOR Operator.....	125
3.5.7.1.1	FOR Operator With the AND Logical Connector.....	129
3.5.7.2	HTOTAL Modifier.....	133

Section		Page
3.5.8	Interfile Output.....	133
3.5.8.1	EXTRACT Operator.....	134
3.5.9	Retrieval Operators.....	138
3.5.9.1	Retireval Type IF Operator.....	138
3.5.9.1.1	FUNCTION Operator.....	145
3.5.9.1.2	Function Work Areas.....	147
3.5.9.2	FIND Operator.....	148
3.5.9.3	Retrieval Operators and Secondary Indexing.....	149
3.5.9.3.1	Retrieval Type IF Operator and Secondary Indexing.....	150
3.5.9.3.2	FUNCTION Operator and Secondary Indexing.....	153
3.5.9.3.3	FIND Operator and Secondary Indexing	153
3.5.10	LIMIT Operator.....	154
3.5.11	KEYWORD Operator.....	155
3.5.11.1	Condition Clauses and Keyword Indexing.....	156
3.5.11.2	Examples of Keyword Indexing.....	157
3.5.12	Sort Operator.....	159
3.5.12.1	Sort Operator With the Sort Space Allocation Modifier.....	162
3.5.13	Library Action Operators.....	164
3.5.13.1	LOAD QUERY Operator.....	164
3.5.13.2	LOAD RIT Operator.....	166
3.5.14	Terminal Environment Operators.....	168
3.5.14.1	SIGNON Operator.....	168
3.5.14.2	SUBFILE Operator.....	171
3.5.14.3	SIGNOFF Operator.....	176
3.5.14.4	TRACE Operator.....	176
3.5.14.5	FORMAT Operator.....	179
3.5.14.6	VIEW Operator.....	180
3.5.14.7	OMQ Operator.....	180
3.6	Field Modifiers.....	180
3.6.1	Periodic Set Control (ALL).....	180
3.6.2	Subroutine Expressions (##).....	181
3.6.3	EDIT Keyword (').....	181
3.6.4	Partial Field Notation (m/n).....	181
3.7	Universal Match Character ($).....	182
3.8	Skeleton Query.....	182
3.9	Noise Words.....	183
3.10	Operator Initiators.....	183
3.11	Logical Connectors.....	184
3.12	Relational Operators.....	184
3.13	Geographic Operators.....	185
3.14	Query Documentation.....	185
3.14.1	NOTE Operator.....	185
3.15	Sample QUIP Queries.....	185
3.16	Operator Summary.....	190
3.17	QUIP Definitions.....	198

Section		Page
4	SOURCE DATA AUTOMATION (SODA).....	202
4.1	Language.....	202
4.1.1	Input Line Formats.....	202
4.1.2	Keywords.....	203
4.1.2.1	FILE.....	203
4.1.2.2	REPORT.....	204
4.1.2.3	CANCEL.....	204
4.1.2.4	UPDATE.....	204
4.1.3	Value Words.....	205
4.1.4	Elements of an Input Request.....	205
4.1.4.1	File Modification Element.....	205
4.1.4.2	Execution Request Element.....	206
4.1.4.3	Disposition Element.....	206
4.2	Input Using a Local 2260.....	206
4.3	Corrections to Terminal Transmissions.....	208
4.3.1	Character Substitution Capability...	208
4.3.2	Standard Line Replacement Capability.....	212
4.4	Output Messages and Actions.....	213
4.5	Restrictions.....	214
4.5.1	File Access.....	214
4.5.2	Transaction Input.....	215
4.5.3	Logic Statements.....	215
4.5.4	Error Reporting.....	215
4.6	Auxiliary Operations While Using SODA.....	215
4.7	Step-by-Step Example of SODA Execution.....	216
5	EDIT.....	220
5.1	General Operation.....	220
5.2	General Concepts.....	221
5.2.1	Terms Used in EDIT.....	221
5.2.2	SEQ/NOSEQ Mode.....	222
5.2.3	Modification of Retrieval Data.....	222
5.2.4	Permanent Data Sets.....	222
5.2.5	Temporary Data Sets.....	223
5.3	Language Description.....	223
5.4	Commands.....	224
5.4.1	Library Name Default Option.....	225
5.4.2	Line Number Indexing.....	225
5.4.3	Zero Line Numbers.....	225
5.4.4	Command Description Conventions.....	226
5.4.5	Control Commands.....	226
5.4.5.1	ALLOCATE Command.....	226
5.4.5.1.1	ALLOCATE Command Parameters.....	227
5.4.5.1.2	ALLOCATE Command Restrictions and Considerations.....	229
5.4.5.2	FORMAT Command.....	229

Section		Page
5.4.5.2.1	FORMAT Command Parameters.....	229
5.4.5.2.2	FORMAT Command Restrictions and Considerations.....	231
5.4.5.3	Control SIGNON Command.....	231
5.4.5.4	SIGNOFF Command.....	232
5.4.5.4.1	SIGNOFF Command Parameters.....	232
5.4.5.4.2	SIGNOFF Command Restrictions and Considerations.....	232
5.4.5.5	SUBMIT Command.....	233
5.4.5.5.1	SUBMIT Command Parameters.....	233
5.4.5.5.2	SUBMIT Command Restrictions and Considerations.....	233
5.4.5.6	UTIL Command.....	233
5.4.5.6.1	UTIL Command Parameters.....	234
5.4.5.6.2	UTIL Command Restrictions and Considerations.....	235
5.4.5.7	VERIFY Command.....	235
5.4.5.7.1	VERIFY Command Parameters.....	235
5.4.5.7.2	VERIFY Command Restrictions and Considerations.....	236
5.4.6	Input Commands.....	236
5.4.6.1	GET Command.....	237
5.4.6.1.1	GET Command Parameters.....	237
5.4.6.1.2	GET Command Restrictions and Considerations.....	238
5.4.6.2	Input MOVE Command.....	239
5.4.6.2.1	Input MOVE Command Parameters.....	239
5.4.6.2.2	Input MOVE Command Restrictions and Considerations.....	241
5.4.6.3	SETUP Command.....	241
5.4.6.3.1	SETUP Command Parameters.....	242
5.4.6.3.2	SETUP Command Restrictions and Considerations.....	244
5.4.6.4	Input SIGNON Command.....	244
5.4.6.4.1	SIGNON Command Parameters.....	244
5.4.6.4.2	SIGNON Command Restrictions and Considerations.....	245
5.4.7	Modification Commands.....	246
5.4.7.1	CHANGE Command.....	246
5.4.7.1.1	CHANGE Command Parameters.....	246
5.4.7.1.2	CHANGE Command Restrictions and Considerations.....	248
5.4.7.2	Modification DELETE Command.....	249
5.4.7.2.1	Modification DELETE Command Parameters.....	249
5.4.7.2.2	Modification DELETE Command Restrictions and Considerations.....	250
5.4.7.3	INSERT Command.....	250
5.4.7.3.1	INSERT Command Parameters.....	250
5.4.7.3.2	INSERT Command Restrictions and	

Section		Page
	Considerations	251
5.4.7.4	RESEQ Command	251
5.4.7.4.1	RESEQ Command Parameters	251
5.4.7.4.2	RESEQ Command Restrictions and Considerations	251
5.4.8	Output Commands	251
5.4.8.1	Output DELETE Command	252
5.4.8.1.1	Output DELETE Command Parameters	252
5.4.8.1.2	Output DELETE Command Restrictions and Considerations	253
5.4.8.2	HOLD Command	253
5.4.8.2.1	HOLD Command Parameters	253
5.4.8.2.2	HOLD Command Restrictions and Considerations	254
5.4.8.3	LIST Command	254
5.4.8.3.1	LIST Command Parameters	254
5.4.8.3.2	LIST Command Restrictions and Considerations	257
5.4.8.4	Output MOVE Command	257
5.4.8.4.1	Output MOVE Command Parameters	258
5.4.8.4.2	Output MOVE Command Restrictions and Considerations	258
5.4.8.5	PUT Command	259
5.4.8.5.1	PUT Command Parameters	259
5.4.8.5.2	PUT Command Restrictions and Considerations	260
5.5	Summary of Commands	261
5.6	Sequence Matrix for EDIT Component Commands	267
5.7	Terminal Response Messages	268
5.8	Terminal Session Example	269
5.8.1	Generate and Error-Scan a Logic Statement	270
5.8.2	Correct and Error Scan the Logic Statement	272
5.8.3	Retrieve and Modify a Second Logic Statement	274
5.8.4	Retrieve, Modify, and List Required JCL	275
5.8.5	Update the Library, Submit the Job, and Sign Off	277
5.9	EDIT Component Reserved Words	278
5.10	EDIT Component Glossary	279
5.11	User Written Format Subroutine	281
5.11.1	Interface Specifications	281
5.11.2	Parameter Area	282
5.11.3	Consequence of Return Codes	282
5.11.4	Interfacing with EDIT	283
5.12	Verification of ALC Source Code	286
5.12.1	Assembler Options	286

Section		Page
5.12.2	ALC Verification Results.....	287
6	FORMATTER.....	289
6.1	General Description.....	289
6.2	General Concepts.....	289
6.2.1	Related Terms.....	289
6.2.2	Definition and Operational Phases...	292
6.3	Format Controlled Access.....	293
6.4	Definition Phase.....	293
6.4.1	Format Definition Statements.....	293
6.4.1.1	DISPLAY Statement.....	294
6.4.1.2	FIELD Statement.....	296
6.4.1.3	General Coding Procedures.....	301
6.4.2	Entering the Format Definition on the Library.....	301
6.4.3	Examples.....	302
6.4.4	Dynamic Specification of Formats....	308
6.4.4.1	FMSODA Format Specification.....	308
6.4.4.2	FMSODA Parameter Specification.....	309
6.4.4.3	QUIP Format Specification.....	310
6.4.4.4	Example.....	311
6.4.5	Restrictions.....	314
6.5	Operational Phase.....	314
6.5.1	Display Session.....	314
6.5.1.1	Initialization.....	314
6.5.1.2	Format Name Command.....	315
6.5.1.3	UPDATE Command.....	319
6.5.1.4	GETDATA Command.....	319
6.5.1.5	SIGNOFF Command.....	320
6.5.1.6	Display Action Code.....	320
6.5.2	Error Procedures.....	320
6.5.3	Example of Terminal Session.....	321
6.6	System Flow Summary.....	329
	DISTRIBUTION.....	332
	DD Form 1473.....	336

ILLUSTRATIONS

Figure		Page
1	2260 Page Display.....	40
2	Sample Output.....	47
3	Sample Page Display 1.....	48
4	Sample Page Display 2.....	49
5	Sample Page Display 3.....	50
6	Sample Display Generated by LIST When Input Message Queue Contains Data.....	52
7	Sample Display Generated by LIST When Input Message Contains No Data.....	53
8	MTO Illustration.....	55
9	Display of Operator to Terminal Message..	57
10	Format Subroutine Interface.....	284
11	Sample Format Subroutine.....	285
12	Output from a Successful ALC Verify.....	288
13	Sample Output from an Unsuccessful ALC Verify.....	288
14	System Overview.....	291
15	Sample Display Format.....	303
16	Sample FORMATTER Definition.....	304
17	Sample Format with Lightpen Selectable Fields.....	306
18	Sample FORMATTER Definition with Lightpen Selectable Fields.....	307
19	Introductory CRT Display.....	316

20	UPDATE CRT Display.....	317
21	GETDATA CRT Display.....	318
22	Definition Phase Overview.....	330
23	Operational Phase Overview.....	331

TABLES

Table		Page
1	Function of 2260 Display Station Control Keys.....	8

ABSTRACT

This volume defines the Terminal Processing (TP) component of the NIPS 360 FFS. It describes the on-line capabilities and language specifications for effective use of the component. The TP Monitor/Supervisor and specific TP Application Programs are discussed. This document supersedes CSM UM 15-74, Volume VI (TP).

CSM UM 15-78, Volume VI, Terminal Processing (TP) is part of the following additional NIPS 360 FFS documentation:

CSM UM 15-78	Vol. I	- Introduction to File Concepts
	Vol. II	- File Structuring (FS)
	Vol. III	- File Maintenance (FM)
	Vol. IV	- Retrieval and Sort Processor (RASP)
	Vol. V	- Output Processor (OP)
	Vol. VII	- Utility Support (UT)
	Vol. VIII	- Job Preparation Manual
	Vol. IX	- Error Codes
TR 54-78		- Installation of NIPS 360 FFS
CSM GD 15-78		- General Description

TR 54-78, Installation of NIPS 360 FFS, is referred to throughout the text as "Installation Document."

TERMINAL PROCESSING (TP)

Section 1

INTRODUCTION

This volume of the User's Manual describes the Terminal Processing (TP) component which provides the system interface to terminal devices. Supported devices are locally attached IBM 2260 Display Stations and IBM 2250 Display Units, remote IBM 2260 Display Stations, IBM 1050 dial-up (switched network) terminal systems, IBM 3270 Information Display System, and IBM 2741 (nonswitched) terminals. This component permits utilization of these devices in a true on-line mode, allowing different tasks to be initiated from the devices without scheduling the tasks through the normal OS/360 job stream.

The TP component consists of three main functional programs; the TP Monitor, the TP Supervisor and the TP Driver. The TP Monitor services device requests, performs input/output queuing and controls task assignments. The TP Supervisor primary function is to provide an interface with TP application programs. The TP Driver initiates both the TP Monitor and Supervisor and has them execute within a single region/partition of core.

To provide better understanding of the functional role performed by TP Monitor/Supervisor, it is necessary to consider the use of a TP application program, such as the Quick Inquiry Processor (QUIP). In typical use of the TP component, a user enters, from a terminal, lines of information which comprise his inquiry. The Monitor accepts these lines (as well as lines from other devices) and queues them on the Input Message Queue. When the terminal operator has completed the formulation of his inquiry, he enters a "command" which requests the Monitor to initiate the QUIP processor. The Monitor recognizes the command, initializes a nonbusy Supervisor which is eligible to service the device

and QUIP, and returns to a wait state. The Supervisor causes the QUIP processor to be loaded and execution to be initiated. QUIP, as one of its first steps, requests the input queue for the device from which the request for QUIP originated. The Monitor provides the original user data to QUIP for processing. During processing QUIP forms output lines representing the information requested by the user, and passes these lines to the Monitor for output queuing. Although reference is made to the Monitor, QUIP actually presents all of its requests to the Supervisor, which is the interface provided for application programs.

When output is available on the output queue, the user may request this information to be displayed. These requests are serviced by the Monitor, which includes within its organization a function called PAGE. PAGE permits display of data from an output queue at a user terminal. The terminal user is permitted to page forward and backward, "roll" his page image forward or backward by line, or request the "nth" line of his output queue to be displayed.

Input requests can be received from all terminals concurrently; output can be displayed at all terminals concurrently. Multiple tasks per Supervisor are supported in both MFT and MVT environments, limited only by core space. Again using QUIP as an example, the TP application task may only be used by a single terminal at a time for each occurrence of the task in the core memory of the computer. However, his "busy" period is limited to the actual task time, and is not related to the input and output queuing function provided by the Monitor.

The TP component is a normal evolutionary upgrade of the Remote Device component it replaces, and as such was designed as a generalized graphics support program. Requests from display terminals are generally processed by dynamically executing TP application programs fetched from the Program Library to respond to the requests. This concept provides for unlimited expansion of the graphics capability. TP application programs can be written and added to the Program Library without any modification to the existing TP component modules. Thus, the functions of the

TP component can be tailored to meet the needs of each installation.

Since many facilities already possess graphic programs of various types, an extension of capability that permits use of these existing "stand alone" programs under the Monitor has been provided. Although use of this feature does cause some redundancy of core utilization, it provides a significant method of quickly and effectively utilizing existing software without reprogramming. In many instances, only minor modifications of the existing software will permit elimination of redundant core penalty.

Section 2 of this volume describes the general terminal operating procedures applicable to the TP Monitor/Supervisor; subsequent sections describe the individual TP application program procedures and requirements.

Section 2

TERMINAL PROCESSING (TP)

In the subsections which follow, certain terminology has been used to refer to specific items, events, and conditions which are unique within the TP component. These terms are defined below:

a. TP Monitor Partition (MFT environment) or Region (MVT environment) -- Any core area under the 360 Operating System in which the TP Monitor has been initialized.

b. TP Application Program -- Any program written specifically to run under control of the TP Monitor.

c. Input Message Queue -- A disk data set or core resident area which all terminal inputs are queued.

d. Output Message Queue -- A disk data set on which TP Application program output is queued for terminals.

e. Input Message -- One "line" of input from a terminal.

f. Input Line -- Same as input message.

g. Request -- One or more input messages constituting all input, and the call for a particular TP Application program.

h. Query -- A request for QUIP containing logical retrieval and output statements.

i. TP Device -- A 2250, 2260, or 3270 Display Station or 1050, or 2741 terminal.

j. Device State -- The current mode of operation assigned to a display station. There are three states which a station may be in. The device state determines the type of input which will be accepted at that time.

- o Active State -- Currently processing a request.
- o Wait State -- Conversational program initiated and currently waiting for input from the device.
- o Inactive State -- Not active or not in the wait state. The device is receiving problem program input or is not being used.

k. Display Command--Input to the conversational program PAGE: required to direct the output display.

l. Page -- One device or screen "loader."

m. Output Line--A single line (1 row) of data on the terminal.

n. Logical NOT Symbol (¬) -- A special character used to indicate the start of a TP control message. It is generally not printable and will be referenced throughout this section as (logical NOT).

2.1 Supported Terminal Devices

The current system release supports locally attached 3270 and 2260 Display Stations and 2250 Display Units. Remote 3270 and 2260 Display Stations, 1050 terminal systems, and 2741 terminals are supported as well. The following sections provide an abbreviated user guide for the four device types.

2.1.1 IBM 2260 Display Station Operation

A complete description of the 2260 Display Station and operating instructions can be found in the IBM Systems Reference Library Form A27-2700, IBM System/360 Component Description, IBM 2260 Display Station, IBM 2848 Display Control. For convenience, a brief summary of operating instructions are included in this section with a copy of the table "Functions of 2260 Display Station Control Keys" (table 1 of section 2.1) from the above-referenced manual.

2.1.1.1 Display Screen

The display screen or scope has 960 display character positions, 12 rows of 80 characters. When a key on the 2260 keyboard is depressed, a character appears immediately on the scope at the position of the cursor. The cursor appears as a vertical line on the scope below and to the left of the position to which it is pointing. As each character is entered, the cursor moves forward to the next position. The cursor can be moved manually to any position on the scope, with the UP, DOWN, ADVANCE, and BACKSPACE keys.

2.1.1.2 Entering Message

A message can be entered starting at any position on the scope. The START key must be depressed first, followed by the message (from 1 to 80 characters). The message is not transferred until the ENTER key is depressed. Depressing the START key causes the Start-Manual-Input (SMI) symbol (▶) to appear. Depressing the ENTER key causes the End-of-Message (EOM) symbol (▬) to appear. When the message has been transferred, the SMI symbol (▶) will disappear. The keyboard will be locked during the transfer and unlocked as soon as the transfer is complete. As soon as the keyboard is unlocked, the next message can be entered. Before the message is transferred, data on the scope can be modified by setting the cursor to the character(s) and entering new data. The cursor must be reset to the end of the message before depressing the ENTER key. The next input message can start where the previous message ended or can be started on a new line of the scope by depressing the NEW LINE key. This will cause the cursor to be positioned at the first

position of the next line or if already on the last line, it will jump to the beginning of the scope. The entire contents of the scope can be erased at any time by depressing the ERASE key. This also sets the cursor at the beginning of the scope.

2.1.1.3 Printing the Contents of the Scope

If a 1053 printer is attached to the same 2848 control unit as the 2260, the contents of the scope can be printed at any time by depressing the PRINT key on the 2260 keyboard. The contents of the scope are printed starting with the first displayable position up to and including the cursor. The cursor is printed as an exclamation mark. If New-Line (NL) symbols (◀) are on the scope, data between the NL symbol and the end of the line on which it appears is not printed. The NL symbol causes a carriage return-line feed on the printer. Only one 1053 printer can be attached to a 2848 control unit, and it is shared by all 2260s. The print requests are processed serially so that data from different 2260s are not mixed within the printing of a single scope full of data.

If an installation has a TP Monitor generated to recognize the command requesting use of the TPDUMP program, the contents of any of the supported terminal devices may be dumped to any device which is supported by the QSAM access method. (See section 2.5.3, The TPDUMP Program.)

Table 1

FUNCTION OF 2260 DISPLAY STATION CONTROL KEYS (part 1 of 3)

Key	No Shift	Shift
SHIFT	Same	
SPACE/ERASE ADVANCE	Advances cursor one display position with no other modification of the display.	Erases the display position to the right of the cursor and advances the cursor one display position.
BACKSPACE	Backs cursor by one position with no erasures. If cursor is in first display position, it will move to last display position.	Same as No Shift
ERASE DISPLAY (can be operated at any time)	Puts check symbol in cursor position. Cursor moves to the next display position.	Erases entire display and locates cursor in upper left-hand corner of display area (first cursor position).
START UP	Cursor moves up one line on the display. If cursor is already in top line, it will go to the last line. The lateral position of the cursor within the display remains constant.	If no Start MI symbol is displayed, the Start MI symbol is placed in the position the cursor was in when the key was depressed. The cursor is advanced one display position. If a Start MI symbol is displayed when the key is depressed, all display data between the Start MI symbol and the cursor, except data to the right of a new line character,

Table 1 (part 2 of 3)

Key	No Shift	Shift
		is erased. The cursor moves to the first display position following the Start MI symbol
ENTER	Puts check symbol in cursor position. Cursor moves one display position.	Places the EOM symbol (-) in the cursor position. Cursor does not move to next position. All data between the Start MI symbol and the EOM symbol is transferred to the computer via the 2848. If this data contains an NL (new-line) symbol, the data between the NL symbol and the end of the line containing it is not transferred. After a successful message transfer, the cursor remains in its previous position and the Start MI symbol is deleted.
PRINT	Causes a check symbol to be put in the cursor position.	Places the EOM symbol in the cursor position. Causes all data between first display position in first line and the EOM

Table 1 (part 3 of 3)

Key	No Shift	Shift
		symbol, in any line, to be printed by the 1053 Model 1 Printer. If no print device is attached to the 2848, a space is written in the cursor position and the cursor moves to the next position.
NEW LINE DOWN	Cursor moves down one line. If it is already in the bottom line, it moves to the top line. The lateral position within the display remains constant.	Places the NL symbol in the cursor position. The cursor moves to the first display position of the next lower display line. If it is already in the bottom line, it moves to the first display position of the top line. Data between the NL symbol and the end of the line is left undisturbed.
Single-character Keys	Cause symbol to be displayed.	Put check symbol in cursor position. Cursor moves to the next display position.
Double-character Keys	Cause function on lower half of key to be performed.	Cause function on upper half of key to be performed.

2.1.2 IBM 2250 Display Unit Operation

A complete description of the 2250 Display Unit and operating instructions can be found in the IBM Systems Reference Library, Form A27-2701. The keyboard of this device is similar to a standard typewriter keyboard in both character position and function. Only a few general comments are therefore included in this document. These comments are directed towards physical use of the device as an input unit, and are appropriate for use with the current TP application programs provided.

2.1.2.1 Entering Message

A message can be entered starting at any position on the display screen; however, the cursor is normally positioned to the first character position of a line by the system. No START key is required, so the user simply enters the text string desired (1 to 74 characters). The message is not transferred until the user depresses (and holds) the "ALTERNATE CODING" key and then depresses the End of Board (EOB) key. Input of the message is effectively instantaneous, and the cursor is moved to the first position of the next line. The user may immediately initiate entry of his next text string.

2.1.2.2 Cursor Positioning

Unlike the 2260 with the nondestructive cursor, the 2250 space bar causes the characters passed to be erased. The ADVANCE key located at the upper right corner of the keyboard will allow spacing over characters without erasing.

2.1.3 IBM 1050 Dial-Up Terminal System Operation

A complete description of the 1050 terminal system and its operating instructions can be found in the IBM Systems Reference Library, Form A24-3474, IBM 1050 Data Communication System, Principles of Operation. However, a very general summary is included in this document to enable the reader to sign onto the terminal and perform input necessary for executing the TP application programs.

2.1.3.1 Terminal Switches

The configuration of switches on the front panel of a 1050 terminal varies greatly from terminal to terminal. Some common switches and their positions for using TP application programs are listed.

Switches	Position
SYSTEM	ATTEND
PRINTER 1	SEND-REC
KEYBOARD	ON
SYSTEM	not DIAL DISC
TEST	OFF
READER 1	OFF
STOP CODE	SENSE
SYSTEM	PROGRAM
SINGLE CY	OFF
RDR STOP	OFF

2.1.3.2 1050 Sign-On Procedure

The following steps must be taken to sign onto a dial-up 1050 terminal.

a. Press RESET LINE and DATA CHECK buttons.

b. Pick up telephone, press TALK, and dial number.

c. When the computer answers, there will be a "click," a 1-second pause, and finally a beeping tone. Upon hearing this tone, push DATA and hang up.

d. When this is complete, the PROCEED light in the lower right corner of the front panel will light, and the keyboard will unlock. The user must then enter the name of the terminal he is using. For information on naming remote terminals, see the Installation Document, section on "TP Monitor Generation."

If no mistake in the sign-on procedure was made, the terminal is now ready for input. If a mistake was made, the

message "UNABLE TO COMPLETE SIGN-ON PROCEDURE" will be returned. If this appears and if the keyboard is unlocked, repeat step d; otherwise, repeat steps from the beginning.

2.1.3.3 Entering Messages

A message may be entered from any position on the page. No START symbol is required. To send a message, the user need simply type the desired text, depress and hold the "ALTERNATE CODING" (ALTN CODE) key, and then press the "5" key (EOB). The message will be sent immediately. There is no automatic carriage return after input, but the user may press the carriage return at any time after sign-on.

2.1.3.4 Terminal Usage

Several features of using the 1050 terminal for TP application programs should be noted.

a. The logical NOT recognized by the TP monitor is transmitted by pressing and holding the shift key and then pressing the period

b. Uppercase and lowercase letters are indistinguishable

c. Typing error may be corrected by backspacing and retyping

d. The carriage return may be pressed at any time. If the DATA CHECK light should go on, the user should push the DATA CHECK button and reenter the same line. It will take approximately 2 minutes to completely display a page, since the typing speed is 15 characters per second and a full page is 20 lines.

2.1.3.5 Sign-Off

To sign off when finished with the 1050 terminal, the user need simply press the TALK button on the telephone to disconnect the line.

2.1.4 IBM 2741 Nonswitched Terminal Operation

A complete description of the operation of a 2741 Communications Terminal can be found in the IBM Systems Reference Library, Form A27-3001, IBM 2740/2741 Communications Terminal Operator's Guide. Included here, however, is a brief description of the operating procedure for 2741 terminals used in executing TP application programs.

2.1.4.1 Transmission Controls

Only two keys (RETURN and ATTENTION) and a switch are required to control the 2741. The RETURN and ATTENTION keys are located on the keyboard. The Terminal Mode Switch (LOCAL/COMMUNICATE) is located on the left side of the 2741 cabinet.

RETURN key - This key causes the carrier to move to the left margin and the carriage to space the paper. In addition, the RETURN key sends a carrier-return code followed by an end-of-transmission (EOT) code (see "Transmitting and Receiving With a 2741") to the multiplexer if the 2741 is in communicate mode (see Terminal Mode Switch).

ATTENTION key - When in communicate mode, operating the ATTENTION key causes an EOT to be sent to the multiplexer.

Terminal Mode Switch - This switch controls the mode of the terminal. When set to LOCAL, nothing can be transmitted or received from the computer. When set to COMMUNICATE, information may be transmitted along the communications line to and from the computer.

2.1.4.2 Transmitting and Receiving With a 2741

The 2741 terminal operates in either the receive or transmit status. The basic indication of terminal status is the keyboard. The keyboard is locked when the terminal is in receive status and is unlocked when the terminal is in transmit status.

The terminal may be switched from the transmit status to the receive status by sending an EOT to the computer. The EOT may be sent with either the RETURN key or the ATTENTION key whenever the terminal is in communicate mode. The EOT causes the keyboard to lock and the terminal to shift to receive status. If the print element is in uppercase, it will shift down automatically.

In the receive status, messages from the computer may be sent to the terminal. If no data is to be sent, the computer sends an EOT to the terminal causing the terminal to shift to the transmit status and the keyboard to unlock.

2.1.4.3 2741 Sign-On Procedure

The following steps should be taken when signing onto a 2741 terminal:

a. Set Terminal Mode Switch to COMMUNICATE

b. Turn power switch to ON (if power is already on, turn switch to OFF and then back to ON).

c. Press either the RETURN key or the ATTENTION key. This will cause the message "TERMINAL OPEN - ENTER REQUEST" to be typed, after which the keyboard unlocks and the terminal is in the transmit status ready for user input. This message will appear only the first time a terminal is signed on. Thereafter, a single RETURN key or ATTENTION key will be responded to with a carriage space.

2.1.4.4 Entering Messages

A message may be entered from any position on the page simply by typing the desired text. Each character is sent directly to the multiplexer as it is typed - no ENTER or end of block keys are required. Typing errors may be corrected by backspacing and retyping. Should the user wish to delete an entire line, he need only type ¬ ¬ (two logical NOT symbols) followed by a carriage return or ATTENTION.

2.1.4.5 2741 Terminal Usage

The following features of 2741 terminals should be noted when using TP application programs.

a. Those 2741s with keyboards which do not have the (logical NOT) symbol may use the uppercase period (.) in its place.

b. Uppercase and lowercase letters are treated as all uppercase.

c. Typing errors may be corrected by backspacing and retyping.

d. Whole lines of text may be deleted by typing followed by the ATTENTION key or the carriage return key. When the line has been deleted, the carriage will space the paper.

e. The length of text typed on a single line should not exceed 80 characters. If 80 characters are entered, not more than three backspaces should be used for a total of 86 key strokes in addition to the carriage return. After correcting for backspaces, the first 80 text characters will be accepted. More than 86 key strokes will cause wrap-around in the input buffer.

2.1.4.6 Sign-Off

To sign off when finished with 2741 terminal, simply switch the power to OFF.

2.1.5 IBM 3270 Information Display System Operation

A complete description of the operation of a 3270(3277) Display Station can be found in the IBM Systems Reference Library, Form GA27-2742-1, Operator's Guide for IBM 3270 Information Display Systems. Only a few general comments are therefore included in this document. These comments are

directed towards the physical use of the device as appropriate for use with the current TP application programs.

The 3270 Information Display System has three basic components: control unit, display station and printer. The control unit provides the external I/O interface for the 3270 System's attachment to a data processing system. It directs the operation of up to 32 attached display stations and printers. There are two basic control units: one for local applications and one for remote. The display station provides image display of data transmitted from the data processing unit and an attached keyboard enables the user to enter data. There are two basic display stations for the 3270 system: IBM 3277, which is attached to a control unit and has either a 480 character display capacity or a 1920 character capacity, and IBM 3275, which is stand-alone display station which contains its own I/O interface. The printer provides printed copy of data displayed at a display station or data transmitted from the data processing system. There are two basic printers for the 3270 system: IBM 3284, which has a printout rate of 40 characters per second, and IBM 3286, which has a printout rate of 66 characters per second.

2.1.5.1 Special Features

The IBM 3277, Model 2 has 1920 display character positions: 24 rows of 80 characters. Three different keyboards are available: typewriter, data entry and operator console. In addition the keyboard may have program function keys and a selector pen. The selector pen is a light-sensitive pen with which the terminal operator can identify a portion of the displayed message for entry into the application program. The PAGE program which controls displays of the Output Message Queue for a particular display station has been modified to support the light pen and function keys of a 3277 terminal. More information can be found in this document, paragraph 2.1.5.4, about PAGE display commands for a formatted 3277.

The 3277 display station includes the following two data security enhancement features: a keylock and an operator identification card reader. The keylock provides key operated control over communications with the terminal application program. With the key off, the display station will be unavailable to the terminal program and the user will be unable to input messages. The operator identification card reader(also called the badge reader) reads a magnetic card for accounting or security purpose. The text on the card, which can be a maximum of 37 numeric characters is read into the buffer in a nondisplay mode. The data is treated as LOGON information and is passed to TPRECORD for processing. Since the format of the text is customized, the installation must have a validation subroutine generated in the TP Monitor, which can process the data on the magnetic card and determine valid LOGON information. See section 2.2.1 LOGON REQUEST for more information.

Three indicators are found on 3277 display stations which provide a visual indication of the operational status of the entire display station and in particular that individual unit. The indicators are located to the right of the CRT screen in a vertical row and are clearly visual squares of light. The labels that describe each indicator are to the right and parallel. The three indicators and functions are as follows:

SYSTEM AVAILABLE indicator is only an indication that the program in the computer is running and that it will accept interruptions from the display station. If this indicator is not on and one attempts to use the keyboard the INPUT INHIBITED indicator will come on and and data from the keyboard will not be accepted. The RESET key will reset this condition and turn off the INPUT INHIBITED indicator.

INSERT MODE indicator tells that display station is in Insert Mode. The light comes on when the INS MODE key is depressed, and is turned off when the RESET key on the keyboard is depressed. Insert Mode allows the insertion of a character or characters into an existing line without destroying any of the data already there.

The new characters are inserted at the cursor location and the data to the right of that location is shifted right one character position for each character inserted.

INPUT INHIBITED indicator tells that all of the input devices on the display station are disabled. One cannot key any data from the keyboard or select any items with the selector pen. There is no mechanical keyboard lock associated with the indicator to prevent the use of the keyboard, so that the keys on the keyboard can be pressed down normally, but nothing happens.

2.1.5.2 Function of 3277 Typewriter Keyboard

The typewriter keyboard resembles a standard typewriter keyboard in appearance and key layout. The characters that can be displayed consist of 26 uppercase alphabetic characters, 10 numeric characters and 26 symbols and punctuations marks. The keyboard also contains cursor and screen control keys that are required to format and enter an input message.

The cursor control keys provide a means of positioning the cursor without affecting any of the information already on the screen. The cursor control keys are:

UP (↑)	moves cursor up one line step at a time
DOWN (↓)	moves the cursor down one line step at a time
LEFT (←)	moves the cursor 1- character step to the left
RIGHT (→)	moves the cursor 1- character step to the right
BACKSPACE (←)	functions exactly as the LEFT key
TAB (→) and BACKTAB (←)	functions like RIGHT and LEFT keys except they cause cursor movement and is related

to program-defined input fields. In 360 FFS/TP no input field definition is currently made so that the cursor will scan the complete display scope.

NEW LINE (↵) moves the cursor to the start of the next next line

The screen control keys on the 3277 are:

CLEAR blanks every character position on the face of the screen and positions the cursor to the first character position on line 1. While the CLEAR key is depressed the INPUT INHIBITEd indicator is lighted but the SYSTEM AVAILABLE indicator is turned on when the clear operation is completed.

ERASE INPUT with an unformatted screen erases all character locations and moves the cursor to the first character location on the top line on the screen.

ERASE EOF blanks character position in the input field in which the cursor is located. The cursor does not move.

RESET resets the INSERT MODE AND INPUT INHIBITED indicators

INS MODE allows the insertion of a character or characters into the middle of a field without disturbing the information that is already displayed there.

ENTER indicates to the program that the message is complete and ready to be entered into the computer. While the message is being sent the INPUT INHIBITED indicator is on, and is turned off automatically when the I/O operation has completed.

2.1.5.3 Entering a Message

A message can be entered starting at any position of the display screen, however, the cursor is always positioned to the first character of a line by the system. Unless a specially designed user program is being used, the cursor should not be moved up or down the screen except with the CLEAR key. The message is entered by depressing the ENTER key. Depressing the ENTER key causes the INPUT INHIBITED indicator to be lighted and when the message has been transferred the SYSTEM AVAILABLE indicator is relighted. Input of the message is effectively instantaneous and the cursor is moved to the first position of the next line by the system. The user may immediately initiate entry of his next string.

2.1.5.4 PAGE Display Commands for Formatted 3277

PAGE will support the 3277 light pen function keys and 3284/3286 printers when the FULPAGE option is specified as YES at TP monitor generation. If the FULPAGE option is specified as YES, PAGE will perform a single I/O operation to a local 3270 terminal to display a full screen of data. The 23th line on the terminal will be as follows:

```
____-____NP/1____NL/2____PP/3____PL/4____S/5____H/6____F/7____P/8____
```

The cursor will be placed in the second position of the line to allow keyboard entry of commands. The next eight fields of the last line are selector pen detectable, and will cause the appropriate function to take place when selected. The number following the '/' indicates the function key to be depressed to perform the selected function. The last field of the line is 'P/8', if this field is selected with the selector pen or function key 8 is depressed it will cause the current page to be printed on the 3284/3286 printer associated with the 3277.

2.1.6 General CRT Terminal Usage

Following are some features helpful to the CRT console user as he gains experience.

a. Typing errors can be corrected on both consoles by backspacing and retyping.

b. The 2260 and 3270 cursor must be positioned following the last character when ENTER is typed; the 2250 cursor position is not important when EOB is typed nor is the carriage position on the 1050 terminal.

c. To type a blank on the 2260, depress SHIFT and depress the SPACE bar; on the 2250 and 3270, no SHIFT is needed.

d. To skip over a character on the 2260, depress the SPACE bar; on the 2250, depress ADVANCE (upper right corner).

e. To delete all input lines and start over; type a line consisting of ¬S. If QUIP rejects a query, the input is left on the input queue for correction or deletion at the user's option.

f. To replace a line on the input queue, determine the line number (1 is the first line entered). Type a line consisting of nnREPLACEMENT LINE. The line number is always two digits (¬01 for the first line). The replacement line is limited to 77 characters (2260,3270), or 71 characters (2250). For example, suppose the second QUIP line had a misspelled field name. Typing ¬02 LIST UNAME PERS UIC. will change the input queue. In a separate line type ¬Q to rerun the input query. Do not enter the ¬Q with the correction line.

g. Correction of input cannot be done while in paging mode. Either scratch the output or hold it, correct the input and enter ¬P.

h. The time of day may be obtained at any time by typing a line consisting of ¬T.

i. When printing the 2260 screen contents, the cursor stops the printing. When paging, position the cursor following the lines to be printed, depress SHIFT and PRINT.

j. If the keyboard on the 2260 should lock, it may be unlocked by depressing SHIFT and ERASE. On the 2250, depress SHIFT and ALT CODING.

k. On 3270 terminals the 24th line is used to display selected TP status messages. "EOM RECEIVED" and "QUIP STARTED" are messages that appear on the 24th line of the display.

l. Selected TP status messages will cause the audible alarm to sound on alarm equipped 3270 terminals.

2.2 TP LOGON/LOGOFF Procedure

The TP LOGON/LOGOFF procedure is the terminal user interface established to obtain accounting information and terminal utilization statistics for installation use. The terminal user must enter a LOGON request to establish access to the TP system and enter a LOGOFF request to terminate access to the TP system.

The TP LOGON/LOGOFF routines contain the linkage needed to interface with an installation written validation subroutine written by the user, designed to control rightful access to the TP terminal devices and NIPS data files. The user written validation subroutine is optional and is as simple or complex as the installation requires. If the optional installation validation routine is used, it determines the format of the LOGON request.

The terminal user may enter remarks at any time while accessing the TP system. Remarks from the terminal user are recorded with the installation accounting information. Problems or problem areas encountered during access to the

TP system may conveniently be noted with the REMARKS request.

2.2.1 LOGON Request

The LOGON request message initiates a terminal session and provides the TP component with required accounting information. User comments may be included but are optional. For non-3270 terminals LOGON request message is entered as a single line not exceeding 80 characters and must be followed by (logical NOT) R on the same line. The LOGON request message in a TP environment in which no optional installation validation subroutine is used, has the following format:

LOGON (ACCT #, USER I.D.) User Comments (logical NOT) R

Where:

LOGON	Request type; required. This must be first within the request message and may be preceded with or followed by blanks.
(ACCT #, USER I.D.)	Accounting information; required. This must appear second within the request message and may be preceded with or followed by blanks. Both account number and user ID are limited to 10 characters and must be enclosed within parentheses. Blanks should not appear within the parentheses.
User Comments	Self-explanatory; optional. May be used for additional installation accounting information.

(logical NOT) ¬R	TPRECORD program request; required.

The LOGON request message in a TP environment which has an installation validation subroutine, has the following format:

LOGON any data format (logical NOT)¬R

where:

LOGON	Request type: required. This must be first within the request message and may be preceded with or followed by blanks.
Any data format	Text data may be a maximum of 72 characters, required. The format is determined by the installation validation subroutine. The data format of the information read by the 3277 operator identification card reader is a maximum of 37 numeric characters. The text is broken down into fields, and blanks are inserted based upon the presence of field separator control characters on the magnetic strip.
(logical NOT)¬R	TPRECORD program request, required.

2.2.1.1 LOGON Requirements for 3270 Terminals

When a 3270 terminal is initialzied by the TP component the following message is displayed:

```
TERMINAL  NOT LOGGED ON, SPECIFY LOGON INFORMATION.
LOGON
```

The 3270 cursor is positioned following the word "LOGON" in the second line of the LOGON message shown. The entry of the required logon information follows the format described in section 2.2.1 with the following modifications:

a. The word "LOGON" must not be entered

b. The logon information will not be displayed on the terminal screen as it is entered from the keyboard

c. A maximum of 74 characters may be entered as logon information.

d. The ¬R is still required at the end of the logon information.

2.2.2 LOGOFF Request

The LOGOFF request message terminates the terminal session. The terminal operator may include comments to be recorded with the TP accounting and statistical data. The LOGOFF request message must be followed by (logical NOT) R on the same line. The LOGOFF request message format is as follows:

```
LOGOFF    User Comments          (logical NOT) R
```

Where:

LOGOFF	Request type; required. This must be first within the request message and may be preceded with or followed by blanks.

User Comments	Self-explanatory; optional. May be used for installation accounting information.
(logical not) ¬ R	TPRECORD program request; required.

2.2.3 REMARKS Request

The REMARKS request message is provided to terminal operators for remarks pertaining to the terminal session. These remarks are primarily intended for accurate and descriptive comments on problems or problem areas encountered during the terminal session. The remarks are recorded with the TP accounting and statistical data and can be used by the installation to improve the TP environment. The REMARKS request message is entered as a single line not exceeding 80 characters and must be followed by (logical NOT) R on the same line. The REMARKS request message format is as follows:

REMARKS User Comments (logical NOT) R

Where:

REMARKS	Request type; required. This must be first within the request message and may be preceded with or followed by blanks.
User Comments	Self-explanatory; required.
(logical NOT) R	TPRECORD program REQUEST; required.

2.3 Input Messages

There are three types of input messages:

a. TP application program input

b. Conversational input

c. TP control messages.

Each type is handled in a different way and must be entered at specific times. Problem and conversational input can be distinguished only by their contents and the state of the device at the time of entry. TP control messages can be distinguished by a unique character which must precede every control message. This character graphically represented by the symbol (logical NOT) ¬ is entered by simultaneously depressing the appropriate key(s). No spaces are allowed between the symbol (logical NOT) and the message.

2.3.1 TP Application Program Input

All TP application program input must be entered prior to execution of the program. It is entered in segments or lines each of which may contain a maximum of 80 characters. There is no limit to the number of lines which can be entered other than the physical size of the Input Message Queue data set. All program input is stored on the Input Message Queue. Only one set of input lines may be queued for each TP device. The input must be deleted before the next problem program's input may be entered. When a program comes to a normal completion, the input queue is automatically deleted (except QUIP, which must be deleted by the user). When a problem program terminates for reasons other than normal completion, such as invalid input, or an unavailable data set, the input queue is not deleted. The user then has the option of modifying the input and reexecuting the program or deleting the input so that the device is free to process the next request. Both options are accomplished by entering TP control messages which are described in subsection 2.2.3. An example of problem program input is the following QUIP query:

```
FILE TEST360. CLASS UNCLASSIFIED.
IF CNTRY EQ VS. SUM PERS.
```

Details on QUIP input are contained in section 3.

2.3.2 Conversational Input

Conversational input is entered in direct response to a request for input from a conversational program. The input is limited to one 80-character message.

Conversational input does not affect the Input Message Queue. It goes directly to the conversational program for processing. There is a one-for-one correspondence between input and processing during the conversational mode. The conversational program must be initiated before the input is accepted. It then requests the terminal operator to enter input and waits for it. While waiting for input, requests from other TP devices will be processed until the input is received.

The display program, PAGE, is a conversational TP program. It displays one page of output on the scope for each display command (conversational input message) that is entered. All TP devices can process conversational input concurrently.

2.3.3 TP Control Messages

TP control messages are directed to the TP Monitor for immediate processing. They are identified by the character ¬ (logical NOT) which must be the first character of the message. There are four control messages.

2.3.3.1 Program Request Message (Logical NOT) Program Name

This message causes the specified TP program to be scheduled for execution. The program may be any TP application, conversational, or utility TP program. The message cannot be entered when another TP program is scheduled, or active on the same device.

The program request message must always start with the character (logical NOT) and is immediately followed (no intervening spaces) by the name of the TP program being requested or at least the first letter of the name. If the program request is preceded by problem program input, then the program request need not be a separate input message.

It may be appended to the last problem program input line as long as the total input does not exceed 80 characters. The following two examples illustrate the use of the program request message to process a QUIP query.

Example 1:

```
FILE TEST360.  CLASS UNCLASSIFIED.

IF CNTRY EQ VS. SUM PERS. (logical NOT)QUIP
```

Example 2:

```
FILE TEST360.  CLASS UNCLASSIFIED.

IF CNTRY EQ VS. SUM PERS.

(logical NOT)Q
```

2.3.3.2 Abort Message (Logical NOT)STOP

The abort message has two functions. When the TP device from which the message is entered is inactive, it causes all data on the Input Message Queue for that device to be erased. When the TP device is active, the (logical NOT)STOP message causes an abort indicator to be set which the TP programs must periodically interrogate to determine if an abort has been requested. Once a TP program has been initiated, it cannot be terminated by the TP Monitor. The TP program must terminate itself when and if it discovers that an abort has been requested. The abort message may be entered in the form (logical NOT)STOP or (logical NOT)S. It may be entered as a separate input message or may be included in a problem program input line. It is a good practice to enter the abort message before beginning a new input request. This insures that any previous request is erased from the Input Message Queue.

2.3.3.3 Correction Message (Logical NOT)nn

The correction message allows the user to replace any input line on the Input Message Queue without having to delete the entire queue and reenter all new input.

Corrections can be made whenever the TP device is inactive. While entering problem program input, or after a problem program has terminated due to an input error, the user may correct the input, line-by-line, and then enter the program request message to initiate processing of the input. The correction message must be entered as a separate input message. nn is the number of the line to be replaced on the Input Message Queue. Lines are numbered sequentially starting with 1. The data to be placed on the message queue must start in the fourth character position of the message. Blanks are valid characters. Assuming that five lines of data have previously been entered, the following examples illustrate correction message formats:

(logical NOT)01REPLACEMENT DATA FOR LINE 1

(logical NOT)03REPLACEMENT DATA FOR LINE 3

Invalid correction messages:

(logical NOT)5REPLACEMENT DATA (data cannot start in the 3rd character position)

An invalid correction message will cause the message

INVALID CORRECTION MESSAGE

to be displayed on the terminal.

2.3.3.4 Time Request (Logical NOT)TIME

The time request message may be entered at any time. It will cause the time of day to be displayed on the scope in hours (H), minutes (M), and seconds (S) in the form:

TIME HHMMSS.

The time request message may be entered as (logical NOT)TIME or (logical NOT)T. The time display may be erased from the scope by the next output message.

2.3.4 Input Message Summary

The chart below shows each type of TP message, the action it causes, and when it may be entered.

TP MESSAGE	ACTION TAKEN ACCORDING TO DEVICE STATE		
	INACTIVE	ACTIVE	WAIT
(logical NOT) PNAME	Execute TP Program PNAME	Invalid	Invalid
(logical NOT) nn NEW LINE	Replace Line nn on Input Message Queue with NEW LINE	Invalid	Invalid
(logical NOT) STOP	Delete Contents of Input Message Queue for this Device	Set Abort Request Indicator	Invalid
(logical NOT) TIME	Display Time Time on CRT	Display Time on CRT	Display Time on CRT
One line of data, any valid input character except (logical NOT)	Add Message to Input Message Queue	Invalid	Respond to Conversational Input Request

2.3.5 Responses To Input Messages

Occasionally, the TP Monitor will respond to an input message by displaying a status message. These messages and their meanings are listed below.

2.3.5.1 Normal Progress Messages

TERMINAL OPEN - ENTER LOGON REQUEST -- User may enter data

TERMINAL NOT LOGGED ON, SPECIFY LOGON INFORMATION. LOGON -- The 3270 terminal uses must enter LOGON information before any input message or program request.

TERMINAL NOT LOGGED ON, INPUT REJECTED -- User has attempted to use the terminal without having first entered a LOGON request. User should enter the LOGON request before any input message or program request.

EOM RECEIVED -- User has entered a valid ¬X program request and should wait for it to be processed.

QUIP STARTED -- QUIP or other requested problem program has been attached. User should wait for next message (below).

START CONVERSATION -- At least one page of output is ready to be examined. User should enter a line with a paging command.

OUTPUT QUEUE SCRATCHED. CONVERSATION TERMINATED -- User has entered S as a page command. He may now enter another query or correct an erroneous one.

IMQ SCRATCHED -- User has entered ¬S command to scratch the Input Message Queue.

2.3.5.2 Warnings and Errors

2.3.5.2.1 While Paging Through Output

INVALID PAGE COMMAND. RE-ENTER NP, NL, PP, PL, ØØØ1, S, H -- User has not used one of the listed options. He should enter a line containing one of the options.

REQUESTED PAGE NOT YET GENERATED -- QUIP or other output program is still running, and the user has examined the

last available page of output. He should wait 15 seconds, and retry his paging command. To abort QUIP, enter ¬S (the abort is not instantaneous).

END OF O/P QUEUE REACHED -- QUIP or other output program has finished, and the user has examined the last available page of output. He may enter any line number or PP to reexamine any part of the output, or may enter S or H.

OUTPUT QUEUE SAVED. CONVERSATION TERMINATED -- User has entered H as a page command. To resume paging at a later time, enter ¬P. At this time, the user may send a message to the 360 operator, or perform other duties.

OUTPUT QUEUE EMPTY. CONVERSATION TERMINATED -- User has requested page with ¬P but there is no output to examine.

PROBLEM PROGRAM IN PROCESS. UNABLE TO MEET REQUEST -- User has entered S or H as a paging command while QUIP or other output program is still running. The paging command is ignored, and the user should wait for the program to complete.

PAGE IS UNABLE TO OPEN THE OUTPUT MESSAGE QUEUE --

This message is not under the users control. When PAGE was invoked the Output Message Queue could not be OPENed. If this condition persists consult the systems programmer.

UNABLE TO SCRATCH DATA SET
or
I/O ERR OCCURRED DURING READ OF O/P QUEUE. RE-ENTER
or
INVALID LINE NUMBER REACHED. RE-ENTER ØØ1 FOR FIRST PG
-- These three messages are not under the user's control; he should enter another paging command.

END OF OUTPUT QUEUE, PROBLEM PROGRAM TERMINATED-------
Further output to the Output Message Queue is stopped. User may page through the Output Message Queue or dump

the Output Message Queue. A record has been written by TPSUP at the end of the Output Message Queue, with the following format---***TERMINATION RECORD GENERATED BY TPSUP AT END OF OUTPUT MSG QUEUE***

2.3.5.2.2 In Response to a ¬X Program Request

LOGON REQUEST ACCEPTED -- User has successfully entered a LOGON request and may now enter input.

LOGOFF REQUEST ACCEPTED -- User has entered a LOGOFF request. The terminal may not be used further without entering a LOGON request.

LOGOFF REQUEST CANNOT BE ACCEPTED BECAUSE TERMINAL CURRENTLY SIGNED ON TO XXXXXX --XXXXXX -- User attempted to LOGOFF while Signon programs all still active. The Signon programs listed in the message must be terminated before the LOGOFF will be accepted.

REMARKS REQUEST ACCEPTED -- User has entered a REMARKS request. The remarks are recorded with the installation accounting data.

LOGON INFORMATION MISSING OR INCORRECTLY SPECIFIED -- User has entered a LOGON request but did not include or correctly specify the required accounting information. User must enter a correct LOGON request. For a TP environment in which the optional installation validation subroutine is used, this message may be supplemented with an optional message from the installation subroutine that more completely describes the error in the LOGON request.

LOGON INFORMATION MISSING OR INCORRECTLY SPECIFIED-PLEASE RESPECIFY LOGON -- The user of a 3270 terminal has entered a LOGON request but did not include or correctly specify the required accounting information. User must enter a correct LOGON request as described in section 2.2.1.1 (LOGON Requirements for 3270 Terminals). For a TP enivornment in which the optional installation validation subroutine is used, this message may be

supplemented with an optional message from the installation subroutine that more completely describes the error in the LOGON request.

TERMINAL DISABLED UNTIL FURTHER NOTICE -- User has entered a LOGON request which the installation validation subroutine considers to be an error which is gross in nature. The terminal is placed in an inactive status and can not be used again without the console operator's intervention.

NO REMARKS IN REMARKS REQUEST -- User has entered a REMARKS request without including remarks.

INVALID REQUEST -- User has entered a request to the TPRECORD program which cannot be identified as a valid request type, LOGON, LOGOFF, or REMARKS.

ILLEGAL PROGRAM NAME IN REQUEST -- The character following (logical NOT) requests a program that is not available on the library or not available from this terminal. User can request a different program, or use another terminal from which the desired program can be requested.

INSUFFICIENT [DISK SPACE] [CORE], REQUEST NOT RUN -- User should wait 1 minute, and reenter a line with his program request. After repeated responses, request a phone call from the operator.

YOUR STOP REQUEST ACKNOWLEDGED. TERMINAL FREE. -- User has aborted a requested query before it could be started. No input is saved. User may start over on a new query.

ABEND in TP program completion code - SYSTEM =SXXX USER=UXXX ***. Requested program ABENDED. User may page the existing output if the START CONVERSATION message follows. If not, there is no output, and the user should reenter the same or different query. This message cannot appear in an MFT system, but will normally be replaced by the following message when the TP Supervisor is reloaded. In any case, a dump was

taken if a SYSUDUMP DD card was included. In an MVT system, the dump is not printed until the TP Supervisor job is terminated.

TERMINAL RESTARTED - ENTER REQUEST -- Meaning and options are the same as above (ABEND IN TP PROGRAM). This message can appear in either MFT or MVT.

TERMINAL READY FOR ANOTHER REQUEST -- User has been operating an independent graphic program and it has completed (normally or abend). The terminal is now in TPS mode, and can be used for a query, message to operator, or another independent graphic request.

2.3.5.2.3 In Response to a Line of Input Other Than Paging Commands

UNIT BUSY, ABOVE INPUT NOT ALLOWED -- User has entered a line after entering a ¬X request. He should wait until the first request is completed, or the START CONVERSATION message appears.

INPUT QUEUE FULL -- User has entered a query that exceeds the assigned capacity of the input queue area. He should enter ¬S to scratch the query, and reenter it placing more words on each line.

IMQ DELETED -- User has entered ¬S before a ¬X program request. His previous input message queue lines, if any, are not retained. He may start over on a new query, etc.

CORRECTION PROCESSED -- User has entered ¬nn (REPL LINE): line number nn was replaced by the user's text.

INVALID CORRECTION MESSAGE -- User has entered ¬nn (REPLACE LINE), but nn was not a valid number. The line was ignored, and the user should reenter the correction line in a valid form. He may enter ¬S and start over.

INV CNTRL MSG, RE-ENTER LINE -- user has entered a ¬X program request, but X is not in the tables of valid

program names. The line was ignored. The user should enter a valid request.

TIME hh mm ss SECS xxx.xx -- User has entered ¬T. The first field is time-of-day, the second is the elapsed time since the most recent previous ¬X program request (if any). If that program has completed, xxx.xx is the total processing time in seconds and hundredths of a second.

I/O ERR, MSG IGNORED -- A data error occurred while reading the user's entry. This is not under his control, and he should reenter the line.

NO DATA ON THE OUTPUT MESSAGE QUEUE -- A request for the TPDUMP program is entered when there is no data on the Output Message Queue.

OUTPUT QUEUE IN HOLD, CANNOT WRITE - A request for a list is entered after the OMQ has been placed in hold.

2.4 Output

The primary output of the TP component is an Output Message Queue on disk containing problem program output formatted for convenient display. Output can be queued for all stations and can be displayed by all stations concurrently. Output cannot be transmitted from station to station. Each device acts independently from the others. After the output from a request has been queued, it can then be displayed, one page at a time by user command. Data displayed on the scope can be transmitted to a 1053 printer if one is attached.

2.4.1 Output Display

The contents of the Output Message Queue, for a particular station, are displayed by the conversational program PAGE. Generally, PAGE is initiated automatically by a problem program when it has placed output on the queue. It can be initiated manually whenever the display station is

inactive by entering the following program request message:

(logical NOT) PAGE or (logical NOT) P

When the PAGE program has been initiated, either automatically or manually, at a display station the message

START CONVERSATION

may then appear and the display commands can be entered. The first line on the queue is always a security classification. It is displayed on each page of output. This line may be blank if the Output Message Queue was generated by QUIP and the file was structured with no classification or the user has inhibited the writing of the security classification (section 3.5.1.2, CLASS Operator). The remaining lines on the queue are either heading or data lines. The entire queue is a sequential list of output lines which are addressed by their position in the list. Line 1 is the first line (Security Classification).

2.4.1.1 Page Display Format (2260 Display Station)

Figure 1 illustrates the format of each page displayed. There are 12 lines or rows of displayed data. The first line contains the security classification if it exits and has not been suppressed by the user. The first line also contains a line count (LINE XXXX OF YYYY) where X is the line number (position on the output queue list) of the first data line displayed. Y is the total number of lines on the output queue. The first data line always appears as the third line of the display. The second display line is reserved for a heading. If there is no heading on the output queue, this display line will be blank. When a heading line is placed on the output queue, it is displayed with all data lines following it until a new heading line is found; then the new heading line is displayed with all subsequent data lines.

```
UNCLASSIFIED                                LINE 0003 OF 0018

Heading Line 1

Data Line A

Data Line B

Data Line C

Data Line D

Data Line E

Data Line F

Data Line G

Data Line H

Data Line I

Data Line J
```

Figure 1. 2260 Page Display

2.4.1.2 Display Commands (2260 Display Station)

Any group of 10 sequential data lines can be selected for display. Each display command causes one page of data to be displayed. The PAGE program then goes into the "wait" state to await the next display command. The data on the output queue cannot be overwritten by another TP program request. The output queue must be released by a display command before another request (for a TP program which uses the output queue) can be entered. There are seven display commands listed as follows:

a. NEXT PAGE--This command causes the next sequential group of 10 data lines to be displayed. When it is used to display an Output Message Queue that was generated by QUIP, this command will cause the display to initially skip the user's query and coverpage, which are the first few lines on the queue, and to start with the first line of formatted QUIP data output. It may be entered in any of the following forms:

NEXT PAGE

N P

NP

N

NEXT

b. NEXT LINE--This command shifts the previous display forward one line. When it is used to display an Output Message Queue that was generated by QUIP, this command will cause the display to start with the first line of formatted QUIP class output. It may only be entered in the following form:

NL

c. PREVIOUS PAGE--Shifts the previous display back 10 lines or up to a new heading line, whichever occurs

first. If lines 20 through 29 were previously displayed, this command would cause lines 10 through 19 to be displayed (assuming no intervening heading lines). If line 15 was a heading line, and there were no other heading lines between lines 16 and 25, then line 15 would become the new heading line and data lines 16 through 25 would be displayed. This command may be entered in any of the following forms:

PREVIOUS PAGE

P P

PP

P

PREVIOUS

d. PREVIOUS LINE--Shifts the previous display back one line. If lines 10 through 19 were previously displayed, the PREVIOUS LINE command would cause lines 9 through 18 to be displayed. It can only be entered in the following form:

PL

e. n --This command can be any number between 1 and 9999 indicating that the next page should begin at line n on the queue. This command gives the user the flexibility of shifting the display to any position of the output queue. The closest heading line preceding line n on the queue will also be displayed.

f. HOLD--This command cancels the conversation mode but holds the output queue, so that no other data can be written on it. This command may be used to hold the output queue so that it can be printed on the 1403 printer by the TPDUMP utility program. TPDUMP releases the output queue after it has been

printed. After entering the HOLD command, the message

OUTPUT QUEUE SAVED.CONVERSATION TERMINATED

will be displayed. This command may be entered as HOLD or just the character H.

g. SCRATCH--This command cancels the conversation mode and releases the output queue. After it is entered, the message

OUTPUT QUEUE SCRATCHED.CONVERSATION TERMINATED

is displayed. The device is now inactive, and the next request can be entered. This command may be entered as SCRATCH or just the character S. A display command can be entered anywhere on the scope. It is entered by depressing the START key, followed by the command itself, and then depressing the ENTER key. After the PAGE program is first initialized (indicated by the message START CONVERSATION) any one of the following commands may be entered to start the display:

NEXT PAGE

Next Line(NL)

n (line number)

If the first two lines on the output queue are a classification and heading line, n can be equal to 0, 1, 2 or 3 to start the display at the beginning of the queue.

If there is no output on the queue, the following message will be displayed:

OUTPUT QUEUE EMPTY.CONVERSATION TERMINATED

In this case, the PAGE program automatically terminates, and the device is free to enter a new request.

When the last line of the output queue is displayed, entering the commands NEXT PAGE or NEXT LINE will cause the message

END OF O/P QUEUE REACHED

to be displayed. When the end of the output queue is displayed, further display can be obtained only by entering PREVIOUS PAGE, PREVIOUS LINE(PL) or n (valid line number).

When using the 2260, the currently displayed page can be printed at any time on a 1053 printer (if attached to the same 2848 control unit as the display station) by depressing the PRINT key. The contents of the display are printed starting at the upper left corner of the scope and ending at the position of the cursor at the time the PRINT key is depressed.

2.4.1.3 Display Examples (2260 Display Station)

Assume that the Output Message Queue contains 18 lines of output which were generated by a problem program other than QUIP. Assume further that these 18 lines of output are as illustrated in figure 2: one classification line, two heading lines and 15 data lines (A-O). Some of the display commands and resulting displays that might be used are shown in the following examples.

a. If the PAGE program was just initialized, indicated by the message START CONVERSATION the command - NEXT PAGE or NL would generate the display shown in figure 1 in subsection 2.4.1.1. The line count 0004 of 0018 indicates line 3 is the first data line on the queue and there are a total of 18 lines.

b. Entering the command NEXT PAGE again will generate the display shown in figure 3. The classification and heading line are repeated. Since the previous display ended with the 12th line of the queue, the next page begins with line 13. When a new heading line is found on the queue (as in this example), the page is displayed partially unfilled. This ensures that data is always displayed with the appropriate heading.

c. Entering NEXT PAGE again would generate the display shown in figure 4. Line 15 becomes the new heading line, and line 16 is the first data line displayed.

d. Entering NEXT PAGE at this point would cause the message END OF O/P QUEUE REACHED to appear since there is no next page when the end of the queue has been reached.

e. If the command PREVIOUS LINE (PL) were now entered, the display shown in figure 5 would be generated. Shifting the display back one line brings the display to a data line (line 14) which comes under a new heading. The new display starts with line 14 and since line 15 is a heading line, the display page ends.

f. Entering the number Ø, 1, 2, or 3 will generate the display shown in figure 1.

Assume instead that the 18 lines of output on the Output Message Queue were generated by QUIP, that lines 2-14 consist of the user's query and the QUIP coverpage, and that the actual first line of QUIP formatted data output is line 16 (Data Line M).

a. If the PAGE program was just initialized, indicated by the message START CONVERSATION, the command NEXT PAGE or NL would cause the PAGE program to skip the user's query and the QUIP coverpage (lines 2-14) and to start the display with line 16 as illustrated by figure 4.

b. Entering the number 0, 1, 2, or 3 will still generate the display shown in figure 1.

TERMINAL PROCESSING (TP)

Line No.	Contents
1	Unclassified
2	Heading Line 1
3	Data Line A
4	Data Line B
5	Data Line C
6	Data Line D
7	Data Line E
8	Data Line F
9	Data Line G
10	Data Line H
11	Data Line I
12	Data Line J
13	Data Line K
14	Data Line L
15	Heading Line 2
16	Data Line M
17	Data Line N
18	Data Line O

Figure 2. Sample Output

TERMINAL PROCESSING (TP)

```
----------------      --------------------------------
UNCLASSIFIED                  LINE 0013 OF 0018
Heading Line 1
Data Line K
Data Line L
```

Figure 3. Sample Page Display 1

```
TERMINAL PROCESSING (TP)


----------------      ---------------------------------

UNCLASSIFIED                    LINE 0016 OF 0018

Heading Line 2

Data Line M

Data Line N

Data Line O
```

Figure 4. Sample Page Display 2

TERMINAL PROCESSING (TP)

UNCLASSIFIED LINE 0014 OF 0018

Heading Line 1

Data Line L

Figure 5. Sample Page Display 3

2.5 Utility Functions

Two utility functions are provided to aid the user. One function (LIST) enables the user to display the current contents of the Input Message Queue. The other function (MTO) transfers a message from the display station to the console operator on the console typewriter. Each of these functions is described in detail in the following two subsections.

2.5.1 List Input Message Queue (LIST)

LIST is a TP problem program which takes the contents of the Input Message Queue and writes it on the Output Message Queue so that it may be displayed by entering display commands. The rules for using LIST are the same as for any problem program. It can be requested any time the display station is inactive by entering the control message:

(logical NOT) LIST or (logical NOT) L.

After the request has been processed, three status messages will be displayed following the program request message and will appear on the scope as follows:

(logical NOT) LIST

EOM RECEIVED

LIST STARTED

START CONVERSATION

Where the third status message appears, enter the display command NEXT PAGE to display the first page.

Figures 6 and 7 show sample displays generated by LIST when the input queue contains data and when the queue contains no data.

```
-------------------------------------------------------------
        UNCLASSIFIED                    LINE 0003 OF YYYY
    CONTENTS OF INPUT MESSAGE QUEUE FOLLOW
Line 1 From Input Message Queue

Line 2 From Input Message Queue
    o
    o
    o
    o
    o
    o
    o
Line 10 From Input Message Queue
-------------------------------------------------------------
```

Figure 6. Sample Display Generated by LIST When Input Message Queue Contains Data

```
------------------------------------------------------------

    UNCLASSIFED                          LINE 0002 OF 0002

NO DATA ON INPUT MESSAGE QUEUE

------------------------------------------------------------
```

Figure 7. Sample Display Generated by LIST When Input Message Queue Contains No Data

2.5.2 Terminal/Operator Communication

Two-way terminal/operator communication is provided by the TP Monitor. The TP Supervisor need not be initialized for this capability and neither the Input Message Queue nor the Output Message Queue is modified by this capability.

2.5.2.1 Message to Operator (MTO)

The MTO function is available at any time the terminal is active. The message is sent by entering it at the terminal and following it on the same line with ¬M. Only a single line message may be sent.

The contents of the input line up to the ¬m, will be displayed on the computer operator's console, preceded by the following identifying message:

MESSAGE FROM TERMINAL XXXXXXXX

When the message has been sent to the operator, an acknowledgment is returned to the terminal:

MESSAGE SENT TO OPERATOR

If no data were found on the input line, nothing would be sent to the operator, and the following message would be returned:

NO MESSAGE FOUND ON INPUT LINE. END MTO.

Figure 8 shows sample displays generated by MTO.

TERMINAL PROCESSING (TP)

Terminal Display:

1. - ENTERED by terminal user

2. - Response from system

1) PLEASE LOAD THE XYZ JOB DECK. ¬M

2) MESSAGE SENT TO OPERATOR

Computer Operator's Console:

2) MESSAGE FROM TERMINAL XXXXXXXX

2) PLEASE LOAD THE XYZ JOB DECK.

Figure 8. MTO Illustration

2.5.2.2 Operator To Terminal

The computer operator may send a message to one, several, or all active terminals. To send a message, the operator replies MSG to the outstanding request that is always issued by the TP Monitor and includes the terminal name or ALL and the message. ALL will cause the message to be sent to all active terminals. A single terminal name or ALL must be separated from the message by one or more blanks or commas. More than one terminal name may be specified by enclosing the name list in parentheses. Names in the list may be separated by blanks or commas. Up to five names may be included in the list. Characters in the reply may be in uppercase or lowercase. Successful transmission of the message will be indicated by the following response to the OPERATOR:

```
MESSAGE SENT TO TERMINAL(S)
```

At the receiving terminal, the message will be preceded by the following line:

```
MESSAGE FROM OPERATOR --
```

If an invalid terminal name is specified or the message format is incorrect, the following error message will be returned, followed by another request for a message.

```
INVALID TERMINAL NAME OR FORMAT IN REPLY
```

Figure 9 shows a sample display generated by MSG.

TERMINAL PROCESSING (TP)

Computer Operator's Console:

1. -ENTERED by operator

2. -Response from system

1) R 00,'MSG TERMINAL 041 YOUR REQUESTED VOL IS MOUNTED'

2) MESSAGE SENT TO TERMINAL(S)

2) 00 TP STANDING REQUEST.REPLY 'ENA', 'DISA', 'MSG', 'PRTY', 'TPS', OR 'PTM'.

Terminal Display:

2) MESSAGE FROM OPERATOR --

2) YOUR REQUESTED VOL IS MOUNTED

If a message is to be sent to all active terminals, the reply would be in the following form:

1) R 00,'MSG ALL TP WILL BE TERMINATED AT 1800'

Figure 9. Display of Operator To Terminal Message

2.5.3 TPDUMP Program

DUMP is a TP program which transfers the contents of the Output Message Queue, a sequential data set, or a member of a partitioned data set (PDS), to any device which is supported by the Queued Sequential Access Method (QSAM). Such devices may include printers, direct access storage devices, magnetic tape storage devices, or card punches. The user may direct TPDUMP to output the OMQ, or other data set, on the device named by a specific DD statement. The TPDUMP program may be used from any of the supported terminal devices provided that the servicing monitor has been generated to recognize the command requesting this program. (See Installation Manual, "TP Monitor Generation."

2.5.3.1 TPDUMP Use

In order to invoke the DUMP program, the user first enters the ¬ (logical not) D. The following message then appears on the display:

SPECIFY INPUT SOURCE AND DESIRED OUTPUT DEVICE

Any or all of the following DUMP control options may be specified in reply to the above message. Any number of blanks or commas may be used to seperate the control option:

```
[ [FROM=insource] [seq1-seq2 ]  [TO=outdevice]      [CONTROL] ]
                  [    END   ]
                  [seq1      ]
```

where:

a. FROM=insource

Specifies the input data set which is to be transferred. Three types of data sets may be accessed by DUMP:

1. The Output Message Queue(OMQ)

Example: FROM=OMQ
This is the default option

(2) A member of a PDS (library)

Example: FROM=YOURLIB(MEMB1)

(3) A sequential data set

Example: FROM=A.SAM.DATASET

b. seq1-seq2
seq1-END
seq1

This parameter specifies a 1- to 3-digit line number or range of line numbers to be selected from the OMQ. Line numbers must be entered in ascending order. A single line is specified by a single line number, a group of lines by a pair of line numbers separated by a dash or slash. A request that the remainder of the OMQ be printed, starting with a certain line is made by specifying END as the second of a pair of line numbers.

Example: 50,75-100,150-END

The above example directs that line numbers 50, 75 through 100, and 150 through the end of the OMQ be selected from the OMQ.

This parameter is only valid when processing the OMQ as the input to TPDUMP.

c. TO=outdevice

This keyword specifies the device which receives the transferred data set. Two types of output device may be specified:

(1) SYSOUT - Causes contents of the input data set to be routed to the default device.

The default device is the device named on the SYSONLIN DD statement in the TP run deck. If there is no SYSONLIN, the program will output to SYSPRINT DD statement. For 3270 display terminals, the first default device is the associated 3284/3286 printer as defined in the QTPDD macro in Monitor generation. If the 3284/3286 printer is being used the following message will be displayed: 'ASSOCIATED 3284/3286 PRINTER BEING USED'. If no associated printer is defined, SYSONLIN DD statement is used first, and then SYSPRINT DD statement.

Example: TO=SYSOUT
This is the default option

(2) DDNAME - Causes contents of the input data set to be routed to the device named on the specified DD statement. The DDNAME must appear in the TP run deck, if not, an error message is written to the terminal and a request for another user reply is issued.

Example: TO=TAPEDD

d. CONTROL

This keyword specifies that a printer carriage control character appears in the first position of each record to be transferred. This character is used by the system printer to control printer spacing and ejection. The CONTROL option is always in effect when the Output Message Queue is being dumped, but it must be specified for non-OMQ data sets.

When a non-OMQ data set is to be transferred, and the CONTROL option is not specified, a printer carriage control character (a blank which directs a standard printer to space one line) is added to the front of each record.

Example: FROM=MYLIB(ALCPGM),CONTROL

This example directs that a member named ALCPGM of a library (PDS) named MYLIB be transferred to the device defined by the SYSONLIN DD statement in the TP procedure (assumed to be a system printer in this example). The first character of each record of the library member ALCPGM will be processed as carriage control information.

This use of the CONTROL option when no valid carriage control information exists may cause the creation of erroneous printer output.

If no control information is supplied to DUMP, an OMQ to SYSOUT data transfer is assumed.

In order to dump the OMQ, the user must save the OMQ by terminating PAGE with the HOLD command. The DUMP program is then invoked with a (logical not) DUMP.

If an error is detected in the control information specified for DUMP, a message will be displayed which directs the user to respecify all DUMP control information, or "S" to cancel TPDUMP.

Records longer than 133 characters will be truncated to that length before transfer.

For partitioned data sets with an undefined record format, a record length of 80 and a blocksize of 800 will be assumed.

The specified input data set (other than the OMQ) must be cataloged prior to DUMP execution.

In no LRECL or BLKSIZE information is supplied on the DD card used by DUMP, (SYSONLIN or SYSOUT) both values will default to 133. The user may wish to override either or both of these parameters. The impact of the carriage control character must be considered when these overrides are specified.

When TPDUMP is used to transfer the OMQ, the first page of that queue will be displayed at the terminal after the specified output device has been accessed. At this time, the user may page through the OMQ. There will be no end-of-job message from the dump program; however, it will not be possible to scratch the OMQ until DUMP is complete.

When TPDUMP is used to transfer a sequential data set or a member of a library, the message 'DUMP HAS COMPLETED REQUEST' will be displayed upon completion of the dump program. If the OMQ does not exist, the user may enter the next TP request. If the OMQ contains data at the completion of TPDUMP, the user will automatically be placed in the paging mode and the first page of the OMQ will be displayed. In this event, the user may page through, SCRATCH or HOLD the OMQ and continue the terminal session.

The Input Message Queue is retained at the completion of DUMP.

2.5.3.2 Simultaneous Operation From Multiple Terminals

The DDNAME used for a DUMP request will be placed in an unavailable status until the requested dump has been completed. A duplicate request for the DDNAME, from another terminal, produces the message, DDNAME CURRENT IN USE. REPLY WAIT OR CANCEL. If the terminal operator replies WAIT, the terminal is placed in a wait state until the requested DDNAME is available. This message appears on the screen --WAITING FOR REQUESTED DDNAME TO BECOME AVAILABLE. If the reply is CANCEL, the following message id displayed --REQUEST FOR DDNAME CANCELED, TPDUMP TERMINATED.

2.5.3.3 DUMP Examples

Example 1

a. HOLD output QUEUE H

b. Initiate DUMP ¬ D

c. Enter the DUMP control information:

1-20,60-END TO=DDCARD1

This control information specifies that lines 1 through 20 and 60 through the end of the OMQ be transferred to the device defined by the DD statement called DDCARD1 in the TP job deck. Since no FROM value was specified, OMQ is assumed to be input source.

a. Initiate DUMP ¬ D

Example 2

b. Enter the Dump control information:

FROM=PDS1(MEMX) TO=TAPEDD1

This control information specifies that a library member called MEMX contained in a library called PDS1 is to be transferred to the DD statement called TAPEDD1.

Example 3

a. Initiate DUMP ¬D

b. Enter the Dump control information:

FROM=SEQ.DATASET.A CONTROL

This control information specifies that a sequential data set called SEQ.DATASET.A be transferred to the SYSOUT device. The SYSOUT

device, assumed to be a printer, will process the first character of each record as valid printer carriage control information.

Example 4

a. Initiate DUMP ¬ D

b. Enter the Dump control information:

FROM=OMQ 10-100

This control information specifies that lines 10 through 100 be transferred from the OMQ to the SYSONLIN DD device.

2.5.3.4 DUMP Errors

If errors are discovered, a self-explanatory numbered message (single digit) will be displayed. Following all error messages, the input command will be displayed. The error number will appear under terms in error. Following the error message(s) the message 'CONTROL ERROR(S)', RESPECIFY CONTROL INFORMATION OR S TO TERMINATE DUMP'. The user should respecify all control information, correcting any errors, or enter 'S' to terminate the DUMP program.

2.5.4 Terminal/Terminal Communication

Two utilities are used to provide communication between two or more terminals supported by the TP Monitor. Communication is provided in three forms. A terminal operator may send a message, the contents of his Input Message Queue (IMQ) or the contents of his Output Message Queue (OMQ). If he is sending a message, it will be written directly to the receiving terminal. If either the IMQ or OMQ is sent, a disk data set will be allocated and the specified queue copied into it. A message will then be sent to the receiving terminal notifying him of this message queue. One or more terminals may be named as receivers; however, only those which are not busy will be eligible to

receive. Any terminals which cannot receive (busy) will be listed in a message back to the sender. The sender can then send his message to these terminals at a later time.

If the sender desires that his message to another terminal interrupt the terminal's activity, he may prefix his request with the word PRIORITY. This will allow his message to be sent to all specified terminals that are operational. Care must be taken in using this option since the message will be sent without regard of the activity at the receiving terminal. This option should be reserved for critical messages.

2.5.4.1 BLAST

This utility is used to send a message IMQ or OMQ to another terminal. This function is initiated by entering the request as one or more lines on the Input Message Queue followed by (logical NOT) B. The request must be the first lines on the IMQ and in the following format:

XXX TO YYYYA,YYYYB, - - -, YYYYYYYC.

XXX This is the specification of what is to be sent. One of the following three terms is allowed:

- MSG - The remaining lines of the IMQ are to be sent directly to the specified terminals.
- IMQ - The remaining lines of the IMQ are to be copied to a temporary data set and a message sent to the receiving terminals informing them of this data set name for later access.
- OMQ - The sending terminal's OMQ is to be copied to a temporary data set and a message sent to the receiving terminals informing them of this data set name for later access.

Only the first character of each of the above terms will be checked by BLAST.

TO — This is a required term

YYYYY — One or more terminal names separated by one or more blanks or commas. This name list must be terminated with a period, and may extend over more than one input line. If all terminals are to receive, the word ALL is substituted for the terminal name and may be the only term following the word TO.

If the IMQ or OMQ is sent, a data set is allocated which will contain the respective message queue for the receiver to access at his leisure. The name of this data set is of the following form:

AAAAA.QHHMM

where

AAAAA	is the sending terminal's name
.Q	is a constant
HHMM	is the current time of day in hours and minutes

If a message is sent, the following prefix line is sent:

* * * * BROADCAST FROM AAAAA

where AAAAA is the name of the sending terminal. The message follows on succeeding lines.

If the IMQ or OMQ is sent, the following notification message is sent:

* * * * BROADCAST FROM AAAAA QUEUE=AAAAA.QHHMM

where AAAAA.QHHMM is as described previously.

If PRIORITY was specified by the sender, the word PRIORITY will be included in the above messages immediately in front of the word BROADCAST.

If BLAST terminates successfully, the message -

BROADCAST DONE

will be sent to the sender's terminal.

If all receive terminals are busy and PRIORITY has not been specified, the message -

ALL TERMINALS BUSY.

will be returned. If some, but not all Receive Terminals are busy and PRIORITY was not specified, the message -

THE FOLLOWING TERMINAL(S) ARE BUSY AND DID NOT RECEIVE THE BROADCAST

will be displayed followed by a list of the busy terminals that did not receive.

2.5.4.1.1 BLAST Errors

If an error is encountered by BLAST, a self-explanatory message will be returned and BLAST will terminate.

If BLAST terminates due to an error, the message:

BROADCAST TERMINATED. NO ACTION TAKEN

will be returned and the IMQ will not be scratched.

If BLAST completes successfully the message:

BROADCAST DONE, INPUT SCRATCHED.

will be returned and the IMQ will be scratched. This message will also be returned if the message was sent to

more than one terminal and at least one but not all were busy.

2.5.4.2 ACCESS

This utility is used by the receiver of an IMQ or OMQ to page through the queue data set. The initial request to ACCESS must be in the following form:

AAAAA.QHHMM ¬A

The meaning of the queue name is described in the BLAST description. If the queue data set can not be found, the following message is displayed.

QUEUE NOT FOUND, ACCESS TERMINATED.

If the queue is found, the data is copied to the OMQ of the receiving terminal. The first page of data will be automatically displayed; subsequent paging commands are identical to those of the PAGE program described in section 2.3.1.2.

2.5.5 UTILITY Program

The Utility program has been designed as a generalized program which can be used to incorporate utility-type functions to aid the user.

The UNDIAL keyword provides the 2260 terminal user operating on a CP/67 with the ability to initiate the disconnecting of the real terminal. Once the other application has been run on the terminal, the 2260 terminal can be reconnected to TP using the CP DIAL command.

The format of the UNDIAL requests is

UNDIAL¬U

The UNDIAL keyword causes the real 2260 terminal to be disconnected from TP. The UNDIAL function also causes a

LOGOFF request to be generated for that terminal. When the terminal is reDIALed to TP the user must LOGON.

The CP keyword allows the user to enter a CP/67 command while running under NIPS TP. When a CP command is recognized, a test is first made to ensure that the program is executing in CP/67 mode. After the CP command is executed, a diagnostic message is written to the terminal. The format for entry of a CP command is as follows:

CP cccc oooooØ(¬)Utility

where

CP	required literal used to identify entry as a CP command.
ccccc	CP command.
ooooo	CP command operand.
Ø	blank required after CP command entry
¬	logical not (¬) symbol
UTILITY	UTILITY (or 'U') to invoke TP Utility program.

The following diagnostics may be received:

INVALID REQUEST	terminal not in CP/67 mode
CP COMMAND EXECUTED	TP Monitor has issued the requested CP/67 command
INVALID CP COMMAND	the CP command was invalid
INVALID OPERAND	the CP command had an invalid operand
CP COMMAND UNSUCCESSFUL	the CP command was issued by TP Monitor, but not executed by the CP/67.

2.6 Independent Graphic Support (IGS)

The TP component provides for support of independent "graphics" jobs, where this term defines a normal OS/360 job that has been written to run as a normal problem program under the operating system. Typically, these jobs have been written using the Graphic Subroutine Package provided under OS, or in one of the higher level programming languages such as PL/1 or FORTRAN using the graphics capabilities of the languages. Any such job may be stored in the system library and called from the terminals associated with TP Monitor. Normal programming conventions are assumed, and systems preparation in the form of the necessary Job Control Language stream associated with the TP Supervisor(s) must have been accomplished. This process is discussed in the Installation Document and is not included in this document. Use of this capability from the viewpoint of the terminal user is simple, and is covered in the following single paragraph.

2.6.1 Terminal Operations Using IGS

The terminal operator is required to enter the name of the IG program he wishes to use as eight characters. This character string is followed by the normal "logical NOT" symbol and the word GRAPHICS or G as shown below:

PNAMEXXX (logical NOT)GRAPHICS

The terminal must be in inactive status when this message is entered. Following acceptance of this command, the graphics application will be loaded and control transferred to the application. All further terminal conversation is controlled by the application and is therefore beyond the scope of this document.

2.7 VIEW

The VIEW program provides the capability to access distribution data sets. A distribution data set contains previously stored output reports which may be generated

after significant update cycles of the data base. In most cases, large important data bases are generally updated once a day and many query output requests can be predicted. Once the output reports are stored on a data set which is accessible by the terminal, the outputs can be selected and viewed at the terminal as required.

The VIEW program formats a list of output report titles (often called a menu) and allows the user to page through the list and select a report to be viewed. Thus users not familiar with NIPS/TP and the QUIP query language can choose an output from the menu and page through it with a few simple commands. This capability makes the data accessible to the decision making personnel. In addition response is almost instantaneous, since no data base search is required.

Currently VIEW can only be initiated by terminal users of IBM 2260 and 3270 terminals. VIEW supports both the light pen and function keyboard of the 3277 terminal and therefore the user need only use the typewriter keyboard miminum number of times.

2.7.1 Creating A Distribution Data Set

A distribution data set contains previously stored output reports which can be selected and viewed at the terminal as required. The distribution data set name may be a maximum of 44 characters and is specified on the MENUSET DD statement in the TP run deck. (The MENUSET DD statement is an addition to the XTP cataloged procedure.) The distribution data set is a Partitioned Data Set; its logical record length must be 80 and its blocksize may be any multiple thereof. It must contain a member named MENULIST, which is made up of titles and descriptions of the stored reports.

The member MENULIST consists of A and B record formats. The A format, or title record is as follows:

```
MMMMMMMMA_________user specified title_______FFFFFFF__nnnnnnnn
1      8 9 10                                63 64  70  73     80
```

where:

```
1-8  MMMMMMMM   - 1-8 character member name, required
9    A          - the record type, must be A
10-63           - the user specified title of the report
64-70 FFFFFFF   - 1-7 character file name, optional
73-80 nnnnnnnn  - sequence numbers, optional
```

The B format, or narrative description record is as follows:

```
MMMMMMMMB user specified narrative description   nnnnnnnn
1      8910                                  70 73     80
```

where:

```
1-8  MMMMMMMM    - 1-8 character member name, required
9    B           - the record type, must be B
10-70            - narrative description of the report,
                   optional
73-80            - sequence numbers, optional
```

An A record may be followed by 0,1,2, or 3 B records. A maximum of 256 A or title records can be stored in the member MENULIST, of one distribution data set.

The contents of the member MENULIST may be modified by OS utility programs and/or NIPS/TP EDIT in the online environment.

The output reports stored on the distribution data set may be generated in the batch and/or online environment.

The following JCL could be used to store a batch QUIP output on a distribution data set named MENUPDS:

```
// EXEC XQUIPSD
//SYSPRINT DD DSN=&&B,DISP=(MOD,PASS),SPACE=(CYL,(1,1)),
//  UNIT=SYSDA,DCB=(LRECL=81,BLKSIZE=810,RECFM=FB)
//SYSIN DD *

          QUIP source statements
```

```
// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&&B,DISP=(OLD,DELETE)
//SYSUT2 DD DSN=MENUPDS,DISP=OLD
//SYSIN DD *
     GENERATE MAXNAME=1,MAXFLDS=1
     MEMBER NAME=name
     RECORD FIELD=(80,2,,1)
```

The following JCL could be used to store the output from an RIT on a distribution data set named MENUPDS:

```
// EXEC XOPSD
//OPLINE DD DSN=&&B,DISP=(MOD,PASS),SPACE=(CYL,(1,1)),
//   UNIT=SYSDA,DCB=(LRECL=133,BLKSIZE=1330,RECFM=FB)
//SYSIN DD *

          OP source statements

// EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&&B,DISP=(OLD,DELETE)
//SYSUT2 DD DSN=MENUPDS,DISP=OLD
//SYSIN DD *
     GENERATE MAXNAME=1,MAXFLDS=1
     MEMBER NAME=name
     RECORD FIELD=(80,2,,1)
```

The following series of steps could be taken in the online environment to store a QUIP output on a distribution data set name MENUPDS:

1. Execute QUIP query
2. Hold the output message queue
3. Scratch the input message queue
4. Signon to EDIT and issue the following EDIT commands:
   a. /MOVE OMQ TO WORK
   b. /PUT MEM=name LIB=MENUPDS NEW COMP=NONE

In addition any SAM data set on disk may be processed by EDIT and stored on a distribution data set. For example, any output from an OS job that can be stored as a SAM file may be later referenced by EDIT. When EDIT is invoked to get the contents of the SAM file and to put it as a member of the distribution data set, an exit to a user written format subroutine may be taken. The user written format subroutine would be designed to compact or expand the existing record length of the SAM file to 80.

2.7.2 Using VIEW with a 3270 Terminal

When VIEW is invoked by specifying (logical not) V, the first display of five report titles automatically appears on the screen; VIEW then waits for user response. The following three lines appear at the bottom of the screen to guide the user:

```
SELECT OUTPUT TO BE REVIEWED,PAGE THROUGH REPORT LIST,OR SIGNOFF
     NP/1  NL/2  PP/3  PL/4  S/5  P/6
     #1/8  #2/9  #3/10 #4/11 #5/12
```

The user may light pen the selection, use the program function keys or enter via the keyboard, the member name, item number (#n), or page commands. In addition, the titles of the reports, which are highlighted as high intensity characters on the screen are light pen detectable.

2.7.3 Sample 3270 Screen Produced by VIEW

```
                                                    ITEM  1 of xxxx
#1  SITUATION REPORT - WEEK ENDING 03/29/74          SITREP  filename
    THIS CONTINUES A SERIES OF WEEKLY REPORTS ON THE
    SITUATION IN etc. etc.

#2  BUDGET REPORT - WEEK ENDING 03/29/74             BUDREP filename
    .........1st line of narrative description.........
    .........2nd line of narrative description.........
    .........3rd line of narrative description.........
```

```
#3   report title                                          name   filename
     ........1st line of narrative description..........

#4   report title                                          name   filename
     ........1st line of narrative description..........
     ........2nd line of narrative description..........
     ........3rd line of narrative description..........

#5   report title                                          name   filename
     ........1st line of narrative description..........
     ........2nd line of narrative description..........
     ........3rd line of narrative description..........
SELECT OUTPUT TO BE REVIEWED,PAGE THROUGH REPORT LIST,OR SIGNOFF
     NP/1   NL/2   PP/3   PL/4   S/5   P/6
     #1/8   #2/9   #3/10  #4/11  #5/12
```

2.7.4 Using VIEW with a 2260 Terminal

When VIEW is invoked by specifying ¬ (logical not) V, the first display of three report titles and associated information appears on the screen. VIEW then waits for a user response; two types of responses may be entered:

Conversational commands to page through the menu, or signoff from VIEW. Valid conversational commands are:

NP	NEXT PAGE	- display next three menu items
N	NEXT	- display next three menu items
NL	NEXT LINE	- display next menu item
PP	PREVIOUS PAGE	- display previous three menu items
PL	PREVIOUS LINE	- display previous menu item
nnnn	ITEM NUMBER	- a 1- to 4-digit number requesting a specific item in the menu data set
S	SIGNOFF	- signoff from VIEW

Request the display of a report from the distribution data set. Two methods exist to request the display of a selected report.

(1) Enter the report name

(2) Enter 1, 2, or 3 to select the report corresponding to that item number on the screen

2.7.5 Sample 2260 Screen Produced by VIEW

```
                                                        ITEM  1 of xxxx
#1  SITUATION REPORT - WEEK ENDING 03/29/74             SITREP  filename
    THIS CONTINUES A SERIES OF WEEKLY REPORTS ON THE
    SITUATION IN etc. etc.

#2  BUDGET REPORT - WEEK ENDING 03/29/74                BUDREP filename
    .........1st line of narrative description.........
    .........2nd line of narrative description.........
    .........3rd line of narrative description.........

#3  report title                                        name   filename
    .........1st line of narrative description..........

#4  report title                                        name   filename
    .........1st line of narrative description..........
    .......2nd line of narrative description...........
    .......3rd line of narrative description...........

#5  report title                                        name   filename
    .......1st line of narrative description...........
    .......2nd line of narrative description...........
    .......3rd line of narrative description...........
SELECT OUTPUT TO BE REVIEWED,PAGE THROUGH REPORT LIST,OR SIGNOFF
    NP/1   NL/2   PP/3   PL/4   S/5   P/6
    #1/8   #2/9   #3/10  #4/11  #5/12
```

Section 3

QUICK INQUIRY PROCESSOR (QUIP)

This section discusses the Quick Inquiry Processor (QUIP), which is designed to give the user a powerful data retrieval capability and a display language for outputting data. QUIP also has the capability to allow for greater flexibility in output display through the use of a stored RIT (Report Instruction Table) generated by the Output Processor (OP) to format the output display.

3.1 QUIP Execution

The execution of QUIP may be Source Direct or Source Retrieval. In the Source Direct mode, QUIP is executed directly against a 360/FFS data file or a non-NIPS data file. In the source retrieval mode, QUIP is executed against a RASP answer set (QRT/QDF) which may or may not have multiple titles.

When QUIP is executed in the Source Direct mode, the user may invoke a file retrieval with a retrieval type IF statement or a FIND statement. If a retrieval is required, QUIP will attempt a Secondary Indexing retrieval.

Secondary Indexing and keyword indexing provide the user capabilities to index a data file by the contents of a field other than the Record ID. The primary purpose of the capabilities are to provide a faster response time for qualifying data records during retrieval. The principles of File Indexing are described in Volume I, Introduction to File Concepts.

In order to initiate keyword indexing, aside from the existence of Index Descriptor Records, the user must include a KEYWORD statement in his retrieval. The Keyword Operator section describes this statement.

Secondary indexing operates only on fixed-length fields. Keyword indexing operates on variable fields, variable sets

and fixed-length alpha fields. Its primary purpose is to enable retrieval of text data based on the presence of keywords in the text. The criteria for these keywords has been defined by the user. For a detailed description of keywords, see Volume I, File Concepts, Keyword Indexing.

The decision indicating whether index processing should be applied to the retrieval logic is based on the existence of Index Descriptor Records in the data file indicating the file is indexed. If the file is indexed, QUIP will activate the Index Processing function to determine if indexing is feasible based on an analysis of the user's retrieval statement. The format of this statement and criteria used in determining if indexing can be utilized are described in sections 3.5.9.1 and 3.5.9.2.

A retrieval based on the results of Index Processing operates in the candidate-access mode. In this mode, the Indexing Executive compiles a list of potential qualifying records called candidates and only the records in this list are accessed and processed by the QUIP Retrieval Phase.

When both keyword and secondary indexing fields produce candidates, the two lists are merged together to form one final candidate list for retrieval. To maintain rapid response time, the keyword indexed fields in a candidate record will not be scanned by the retrieval phase as the data record has already qualified in the keyword processing phase.

The alternate retrieval technique, a retrieval without Index Processing, operates in the file-access mode. In this mode, the QUIP Retrieval Phase must search and process the entire file, except the records excluded by periodic set selection or LIMIT restrictions.

Secondary Index processing can be bypassed by using the INDEX=NO option in the PARM field of the EXEC statement for a query run in the background environment or by including the keyword INDEXNO as the first word of an online query.

3.2 QUIP Operating Environment

QUIP may operate in an 'off-line' background environment or in an 'on-line' terminal environment. In the background environment the input data set may be a SAM file, concatenated SAM file segments, an ISAM file, a VSAM file or a RASP answer set. The normal mode of output is the paging mode. The paging mode formats the output into pages of a specified length. The page is started by a classification line and a space, and it is ended with a space and a classification line. Headers and trailers, if defined, are present on each page; headers are followed by a space and trailers are preceded by a space. The paging mode is the normal mode for all output going to SYSPRINT. In the terminal environment a direct access resident SAM or ISAM data file is required and the normal mode of output is the display mode. The display mode formats the output as one continuous page.

NOTE: QUIP cannot be used to process a VSAM file in the terminal environment.

When Interfile Output (IFO) is used to access records from related files, all secondary files must be either ISAM files or VSAM files and mounted during execution. Up to a maximum of nine different secondary files may be referenced in any query using IFO.

3.2.1 File Analysis and Run Optimization Statistics

QUIP, when executed in the background, gathers and outputs File Analysis and Run Optimization Statistics. The data set name (DSNAME) of this data set must be the data file name suffixed a T. The T is added to ISAM and VSAM names; the S is replaced by T in SAM names. To obtain transaction output, the DSNAME must be cataloged and the user must specify the volume serial (VTRANS) and unit (UTRANS) in the execution procedure. The volume may be any direct access volume.

If the transaction data set exists at execution time, transactions will be added (DISP=MOD). If the data set does not exist, a five trace data set will be dynamically

allocated. The user may change the allocation value by overriding the TRANST DD card space parameter. Transactions are written as fixed length, unblocked, 50-byte records. The format (fixed) and length (50) cannot be changed but the user may change the blocking factor by specifying a DCB BLKSIZE in the TRANST DD card which is a multiple of 50.

If the user specifies a DSNAME (TRANS) in the TRANS DD card, he must supply all parameters required to process the data set. These parameters must conform to the requirements defined above.

The statistics gathered are in the format of transaction records, suitable for input to an FM run to update a file. The information consists of the file name, component name, source module name, count of the executions of the source and the date the source was executed.

The Run Optimization Statistics are initiated through parameters entered in the PARM field of the EXEC card. The breakdown of the statistics details the amount of core used for user subroutines and tables, logic statements, process block, I/O buffers, and access methods. It includes the number of BLDL entries allocated and used, and the number of entries required for each subroutines, tables, and logic statements to reside in core. The amount of core required for each of them to reside in core is also be output. If any are rolled, this information will also be output with the causes for the rolling and the number of times rolled.

Since QUIP queries are unnamed, the query is named as follows:

QYDDDHHM

where:

Q - appears as shown

Y - last character of the year

DDD - 3-character day of the year

HH - 2-character hour

M - first character of the minutes

This Date-Time name is the Date-Time of the execution of the query.

The user is able to enter override parameters for the number of BLDL entries to allocate and for the size of the processing block desired for storage of data records during QUIP processing.

The parameters that may be entered in the PARM field on the EXEC card are as follows:

ROS - indicates that run optimization information is to be gathered and output.

NOROS - no run optimization information is to be output. If no other parameters are used, this should be omitted.

The parameters used to tailor core allocation are as follows:

TCP=nK - the number (n) of 1000 (K) bytes of core requested for process block.

TCB=n - the number (n) of entries to be allocated for BLDL list.

TCS - use the statistics record on the ISAM data file to compute process block size. The parameter cannot be used with the TCP parameter.

For a more detailed description of the capability, see Introduction to File Concepts, Volume I.

3.2.2 Accessing a SAM Data File from a Terminal

A direct access resident SAM data file can be accessed from a terminal if a SAMFILE DD card is included in the TP Supervisor procedure. For a SAM file query from a terminal, the file name includes the ending 'S', e.g., FILE TESTERS.

CLASS ... If QUIP cannot locate the ISAM file requested in a query, it will concatenate an 'S' to the file name and use the SAM file, if it can be located and mounted.

3.2.3 Subfile Function

In the online environment, the subfile function may be used in conjunction with retrieval logic to define increasingly discrete queries so that each new query processes a decreasing number of data records of the file. This is accomplished by allowing the user to create a subfile consisting of selected record keys from the master file. The entries are selected based on conditional expressions in the query. Subsequent queries are automatically directed against the subfile until the user creates a lower level subfile or specifies a different source of input.

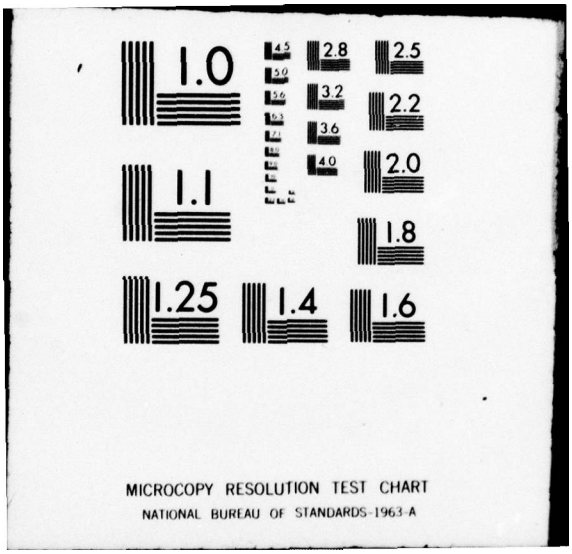
The first time a subfile request is recognized a partitioned data set is dynamically allocated. Each subfile request results in the creation of a member of the partitioned data set. The subfile member consists of the record keys of the qualifying records, rather than the actual records themselves, thereby reducing I/O time and space requirements for processing the subfile. When a query is directed against a subfile, the record keys in that partitioned data set member are examined to determine which records in the master file are to be accessed. If a query which is directed against a subfile contains a retrieval which operates in the candidate-access mode, only those candidates which are found in the subfile are processed by the QUIP Retrieval Phase in examining the candidate list built by Index Processing.

The subfile function allows the user to create a subfile and an output display simutaneously so that the data records of the subfile can be examined while the subfile is being created.

The subfile TRACE function allows a user to review any or all subfiles which were created from the same (current) data file.

AD-A063 431 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 9/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78 C K HILL
UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-6 SBIE-AD-E100 131 NL

2 OF 4
AD
A063431


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

3.2.4 Non-NIPS File Processing

QUIP also has the ability to query data files which were not created by NIPS. These files can take a virtually unlimited number of formats; however, some general restrictions are placed on non-NIPS files. The following are the restrictions which directly apply to the QUIP user:

a. Prior to querying any non-NIPS file, a File Format Table (FFT), describing the format of the file, must be structured as a seperate ISAM or VSAM data set.

b. The non-NIPS file must be in ascending sort order according to the record ID fields in the FFT.

c. The non-NIPS file can be a SAM, an ISAM or a VSAM data set.

d. To be processed, each record ID must contain a non-repeating record. If the nonrepeating record type is not present for a record ID, all the records for that record ID are ignored. The nonrepeating record type will be processed just as the fixed set of a NIPS file is processed.

e. Record IDs may also contain repeating record types which will be processed just as the periodic sets of a NIPS file are processed. Therefore, the rules governing the use of periodic data fields in any of the QUIP operators also apply to data fields from repeating record types in a non-NIPS file.

f. References to the recrod type field will be for the fixed record type only if the name specified for the fixed record type has been duplicated for any other record types. In order to reference the record type field for repeating record types, a unique name for the field must be defined in the FFT for each record type. Then, this unique name must be used to reference the field for the desired repeating record type.

g. Arithmetic operations are limited to those fields idnetified in the FFT as either NUMER or DECML. Numeric data fields which were defined as ALPHA because the format did not conform to NIPS standards, are processed as ALPHA.

h. Secondary and Keyword Indexing is not permitted for non-NIPS files. Also the LOAD RIT operator is not permitted.

A complete description of all restrictions pertaining to non-NIPS files is contained in Volume I - Introduction to File Concepts, Section 3.12.

3.2.5 PBSIZE Paramters

The PBSIZE parameter enables the user to specify the amount of core to be used as a procesing block for the file. The format of the parameter is:

```
PBSIZE  = nnnK
PB
```

where "nnn" is a 1- to 3-digit number specifying the number of 1000 byte segments of core to be allocated as the processing block. When PBSIZE is used, it must precede all QUIP operators. When used in the signon mode, the established processing block size will be used until either a new PBSIZE or a SIGNOFF is encountered. If omitted, a default size is set. For NIPS ISAM files the default size is based on the "N" record and for all other files, NIPS and non-NIPS, the default is 10K.

3.2.6 PBSIZE1 Parameter

The PBSIZE1 parameter enables the user to specify the amount of core to be used as a processing block for all secondary files. The format of the parameter is:

```
PBSIZE1    = nnnK
PB1
```

where "nnn" is a 1- to 3-digit number specifying the number of 1000 byte segments of core to be allocated as the

secondary processing block. This secondary processing block will be used for all secondary files. When used, the PBSIZE1 parameter must precede all QUIP operators. Also, when used in the signon mode, the established secondary processing block will be used until either a new PBSIZE1 or SIGNOFF is encountered. If omitted, a system default size will be used. For NIPS ISAM files the default size is based on the "N" record and for all other files, NIPS and non-NIPS, the default is 10K.

3.3 QUIP Source Language Formatting

The language for this component is free-formatted.

Source statements prepared for a load query card input may begin in any column and continue through column 71. Words and literals may be split between cards if a nonblank character is in the continuation column (column 72). If a card is continued, column 1 of the next card is assumed to immediately follow column 71 of the previous card. This means that when a word is being continued, a blank in column 1 will terminate the previous word; when a literal is being continued, a blank in column 1 will be included in the literal, etc. No constraint is specified on the number of blanks and/or commas separating words in the input stream.

3.4 Output Page Format

The format of an output page without headers or trailers is as follows:

```
-------------------------------------------------------------

                CLASSIFICATION                  DD MMM YY

                     (space)

                USER OUTPUT DATA



                     (space)

                CLASSIFICATION                  PAGE
-------------------------------------------------------------
```

The format of an output page with headers and trailers is as follows:

CLASSIFICATION DD MMM YY

(space)

HEADER 1

HEADER n

(space)

USER OUTPUT DATA

(space)

TRAILER 1

TRAILER n

(space)

CLASSIFICATION PAGE

3.5 Operators Available in QUIP

The operational categories of the QUIP language are as follows:

OUTPUT
- Run Initializing
  - FILE
  - CLASS
- Page Environment
  - HEADER
  - TRAILER
  - PAGENO
  - SYSDATE
  - NLINES
- Imperative
  - LIST
  - DISPLAY
  - MARK
  - PRINT
- Arithmetic
  - SUM
  - COUNT
  - COMPUTE
- Conditional
  - IF
  - FINAL
- Forms Control
  - SPACE
  - EJECT
- Matrix Generation
  - FOR/AND
- Interfile Output
  - EXTRACT

RETRIEVAL
- IF
- FUNCTION
- FIND
- KEYWORD

LIMIT
- LIMIT

SORT
- SORT

LIBRARY ACTION
- LOAD QUERY
- LOAD RIT

FIELD MODIFIERS
- ALL
- Universal Match
- Subroutine Conversion
- Edit Mask
- Partial Field Notation
- Replacement Variable

QUERY DOCUMENTATION
- NOTE

TERMINAL ENVIRONMENT
- SIGNON
- SUBFILE
- SIGNOFF
- TRACE
- FORMAT
- VIEW
- OMQ

In the sections which follow, several symbols are used in describing the format of the QUIP operators. These symbols are not to be coded by the user; they are used only to depict the possible variations of a format. The specific meanings of these symbols are discussed below:

[ ] indicates optional operands. The operand enclosed in the brackets may or may not be coded, depending on whether or not the indicated option is desired. If more than one item is enclosed in brackets, one or none of the items may be coded.

{ } indicates that a choice must be made. One of the operands from the vertical stack within the braces must be coded, depending on which of the indicated options is desired.

(s) indicates that more than one operand of the specified type may be entered. If parentheses appear anywhere else in the format entry, they are a required part of the format and must be entered as shown.

... indicates repetition of a set of operands. The optionally designated set of operands which follows the ellipsis may or may not be coded; if they are coded they may be repeated one or more times.

In addition to these symbols, the following standardized notations are used:

Uppercase options indicate keyword entries and must be coded exactly as shown.

Lowercase options indicate the nature of the entry and must be replaced by the appropriate value.

3.5.1 Run Initializing Operators

The FILE operator identifies the source of input to the query. The CLASS operator identifies either the file password or the security classification of the output from the query.

3.5.1.1 FILE (QUERY) Operator

The FILE (or QUERY) operator is used to identify the source of input to the query which may be a NIPS FFS file, a subfile of a NIPS FFS file, a non-NIPS file, a non-NIPS subfile, or a RASP answer set. When Interfile Output (IFO) is used, the FILE operator identifies the primary file. The FILE operator should be terminated with a period; however, one is inserted by QUIP if it is not present. The format of this operator depends on the input source.

If the input source is a NIPS FFS file or subfile, the format is:

$$\left\{\begin{matrix}\text{FILE}\\ \text{QUERY}\end{matrix}\right\} \left\{\begin{matrix}\text{filename}\\ \text{subfilename}\end{matrix}\right\}$$

a. The operator is either FILE or QUERY.

b. The file name identifies the NIPS FFS file which is to be queried. In the online environment it must be a cataloged data set.

c. The subfilename is a valid operand only in the online environment. It is a 1- to 7-byte name which identifies a NIPS FFS subfile created in a previous online query during the current interrogation session. The subfilename may have been assigned by the user or by QUIP.

If the input source is a non-NIPS file or subfile, the format is:

$$\left\{\begin{matrix}\text{FILE}\\ \text{QUERY}\end{matrix}\right\} \left\{\begin{matrix}\text{filename}\\ \text{subfile name}\end{matrix}\right\} \quad [\text{FFT fftname}]$$

a. The operator is either FILE or QUERY.

b. The file name identifies the non-NIPS data file to be queried. In the online environment it must be a cataloged data set.

c. The subfilename is valid only in the online environment. It is 1- to 7-byte name which

identifies a subfile of a non-NIPS file created in a previous online query during the current interrogation session. The subfilename may have been assigned by the user or by QUIP.

d. FFT fftname identifies the name of the external FFT which describes the format of the non-NIPS file. In the online environment the fftname must be cataloged.

If the source of the input is a RASP answer set, the format is:

$$\left\{\begin{matrix}\text{FILE}\\\text{QUERY}\end{matrix}\right\} \text{ filename retrieval-number [RITname] .}$$

a. The filename is a required entry which follows system naming rules.

b. The retrieval-number is a required entry which identifies the answer set.

c. The RITname is the secondary (RIT) ID which must be specified only if it has been previously identified in the RASP query.

The following conventions are applicable to processing of the FILE operator:

a. No more than one FILE operator is allowed in a query.

b. The FILE operator must be included in any query executed in the background environment.

c. The FILE operator must be included in a query which is executed in the online enviornment unless QUIP is in signon mode (see section 3.5.14, Terminal Environment Operators)

d. When QUIP is in signon mode, the FILE operator is optional. However, it must be included in the first query executed in signon mode to identify the

file to be processed by the current and following queries.

e. When QUIP is in signon mode, any query which does not include a FILE operator uses the same input source as that of the last successful query, providing a subfile was not created by that query (see section 3.5.14, Terminal Environment Operators).

f. When QUIP is in signon mode, the FILE operator may be included in any query to indicate a different input source to be used for the current and following queries.

g. If the subfile function is in use, the FILE operator may be included to specify the master file or any subfile of the master file to override the automatic default whereby a subfile created by the previous query is used as the input source for the following query.

h. If an input source other than the master file or one of its subfiles is specified while the subfile function is in use, any existing subfiles are deleted immediately upon validation of the new input source.

i. If an invalid input source is specified while QUIP is in signon mode, no action is taken. The default input source to the next query is that of the last successful query if that query did not create a subfile. If the last successfuly query did create a subfile, the default input source is that subfile.

Example 1

```
FILE TEST360.
```

Comments: This example specified that the TEST360 file will be input to the QUIP processor.

Example 2

```
FILE TEST360 1.
```

Comments: In this example the TEST360 file is being queried and the input to QUIP is from a RASP answer set labeled 0001.

Example 3

```
FILE TEST360 22 MYRIT.
```

Comments: Same comments as above except that the answer set is labeled 0022MYRIT.

File name may be qualified data set name, in which case the NIPS file name would be the last segment of the fully qualified data set name.

Example 4

```
FILE NIPS.SAMPLE.TEST360.
```

Comments: This sample illustrates the use of a file name as the last segment of a qualified data set name.

Example 5

```
FILE NNFILE FFT NNFFT.
```

Comments: This sample specified that the non-NIPS file NNFILE is to be queried and the format is described by NNFFT.

3.5.1.2 CLASS Operator

The CLASS operator defines the file password or the security classification of the output report. In general, the operator is always required whenever the FILE operator is used if a password or classification has been specified for the file. It should not be used if a password or classification has not been specified for the file. It may

appear only once in the query. The format of the CLASS operator is:

CLASS classification literal [N]

The classification literal is a required entry which may be up to 32 bytes long. It must be enclosed in apostrophes when it contains special characters, blanks, or reserved words. The literal appears centered in the first and last line of every page of output unless the N operand is included. In addition, a coverpage is generated consisting of a page of lines with the literal embedded in a row of asterisks unless the N operand is included.

The character N is specified to inhibit the writing of the classification literal in the first and last line of every page of output and to bypass the generation of the coverpage.

This operator causes the classification supplied to be compared with the classification of the file. If the classifications do not match and the output has been directed to SYSPRINT, an error message appears on the operator console and the output listing, and the run continues with the user-supplied classification. If the classifications do not match and the output has been directed to a terminal, an error message is displayed, and the query is terminated. When IFO is used and the classifications of all the files are not the same, warning messages are displayed, but the query continues. The warning message is not displayed however, if the N character was used to inhibit writing the classification literal.

The use of the "N" operand to suppress the coverpage and the writing of the classification literal in the first and last line of every page of output does not remove the requirement for the operator when the file has a classification, nor does it affect the classification checks above.

Example 1

CLASS UNCLASSIFIED

Comments: A classification of "UNCLASSIFIED" has been assigned to the output report.

Example 2

```
CLASS 'VERY CLASSIFIED'
```

Comments: The classification literal must be enclosed in apostrophes if it contains embedded blanks. "VERY CLASSIFIED" will be the classification of the output report.

Example 3

```
CLASS UNCLASSIFIED N
```

Comments: The writing of the word UNCLASSIFIED in the first and last line of every page of output and the generation of the QUIP coverpage is inhibited. However, the classification match is still performed.

3.5.2 Page Environment Operators

Operators in this category allow the user to place information at the top and bottom of every page of output, to reposition the page number and/or the system date, and to control the number of lines printed on each page. All operators in this category are optional.

3.5.2.1 HEADER Operator

The HEADER operator gives the user the capability to define an output heading. It is followed by one or more literals which need not be enclosed in apostrophes unless they contain blanks, special characters, or reserved words. If multiple literals follow the operator, they will be concatenated into one literal. In the paging output mode, the literal will be centered in a blank line and displayed at the top of each page of output by the output executor. In the display output mode (terminal output), the header(s) will appear only on the first screen of output displayed. In either case, all headers will be followed by one blank line.

If multiple headers are defined, every HEADER operator must be immediately followed by one digit, i.e., HEADER0...HEADER9; hence, the user may define up to 10 headers per QUIP run. If the digit is present, a sequence check will be performed, and any headers that are out of sequence will be flagged. The format of the HEADER operator is:

```
HEADERn   literal
```

a. n is a 1-digit number identifying the HEADER.

b. The literal may not exceed the length of the output line.

Example 1

```
HEADER   'FINAL REPORT'
```

Comments: A digit need not be appended to the operator when only one header is defined.

Example 2

```
HEADER2 'FINAL REPORT'
HEADER4 'OCTOBER 1970'
```

Comments: The digits do not have to be consecutive, but they must be in ascending order.

Example 3

```
HEADER 'THIS REPORT IS DESIGNED TO FACILITATE THE '
       'TABULATION OF WEEKLY AND MONTHLY ACCOUNTS'
```

Comments: If the entire literal cannot be contained on one card, it may be continued on the next card and QUIP will concatenate it into one literal. The user must supply any separating blanks or punctuation within the literal.

3.5.2.2 TRAILER Operator

The TRAILER operator gives the user the capability to place descriptive information concerning the report at the

end of each page of output. It is followed by one or more literals which need not be enclosed in apostrophes unless they contain blanks, special characters, or reserved words. If multiple literals follow the operator, they will be concatenated into one literal. In the paging output mode, the literal will be centered in a blank line and displayed at the bottom of each page of output by the output executor. In the display output mode (terminal output), the trailer(s) will appear only on the last screen of output displayed. In either case, all trailers will be preceded by one blank line.

If multiple trailers are defined, every TRAILER operator must be immediately followed by one digit, i.e., TRAILER0...TRAILER9; hence, the user may define up to 10 trailers per QUIP run. If the digit is present, a sequence check will be performed, and any trailers that are out of sequence will be flagged. The format of the TRAILER operator is:

```
TRAILERn  literal
```

a. n is a 1-digit number identifying the TRAILER.

b. The literal may not exceed the length of the output line.

Example 1

```
TRAILER 'FOR INTERNAL USE ONLY'
```

Comments: A digit need not be appended to the operator when only one trailer is defined.

Example 2

```
TRAILER3 'SUBMITTED'
TRAILER5 'OCTOBER 1970'
```

Comments: The digits do not have to be consecutive, but they must be in ascending order.

Example 3

```
TRAILER 'DISTRIBUTION OF THIS REPORT IS AS FOLLOWS: '
        'PERSONNEL, FINANCE, AND MAINTENANCE DEPARTMENTS'
```

Comments: If the entire literal cannot be contained on one card, it may be continued on the next card and QUIP will concatenate it into one literal. The user must supply any separating blanks or punctuation within the literal.

NOTE: Normal spacing between Headers/Trailers is single space. The SPACE operator should not appear within the range of HEADERS or TRAILERS. Spacing between HEADERS and TRAILERS may be accomplished by inserting a HEADER/TRAILER with a blank literal.

3.5.2.3 PAGENO Operator

The PAGENO operator allows the user to reposition the page number of each page. It must be followed by two literals (UPPER or LOWER and LEFT or RIGHT) describing the location. These literals do not have to be enclosed in apostrophes. Only one use of this operator is permitted per query; if it is omitted, the page number will still be printed. The default position is LOWER RIGHT. When it is output, the page number will appear on the same output line as the classification literal. The format of the PAGENO operator is:

```
                UPPER    LEFT
PAGENO          LOWER    RIGHT
```

Example

```
PAGENO LOWER LEFT
```

Comments: The page number will appear in the lower left corner of every page of output.

3.5.2.4 SYSDATE Operator

The SYSDATE operator allows the user to reposition the system date (DD MMM YY) on each page. It is followed by two literals (UPPER or LOWER and LEFT or RIGHT) describing the

location. These literals do not have to be enclosed in apostrophes. Only one use of this operator is permitted per query; if it is omitted, the system date will still be printed. The default position is UPPER RIGHT. When it is output, the system date will appear on the same output line as the classification literal. The format of the SYSDATE operator is:

$$\text{SYSDATE} \quad \begin{Bmatrix}\text{UPPER}\\ \text{LOWER}\end{Bmatrix} \quad \begin{Bmatrix}\text{LEFT}\\ \text{RIGHT}\end{Bmatrix}$$

Example

```
SYSDATE UPPER LEFT
```

Comments: The system date will appear in the upper left corner of every page of output.

Note: The page number and the system date may not be specified to appear in the same position.

3.5.2.5 NLINES Operator

The NLINES operator gives the user the capability to force the page or display mode of output. In normal operations, the NLINES operator forces the paging mode of output and is followed by a numeric literal specifying the number of lines to be printed per output page. A special indicator of Ø, however, may be used to force the display output mode. The user must place this operator early in the source deck so that the new number of lines will become effective before the first page of source listing exceeds it.

The default options provided for the NLINES operator are dependent on the output mode. If the output mode is paging, the default value is 60. If the output mode is display, the default value is Ø.

The method of calculating NLINES is:

```
NLINES=number of lines available for output
       + 4 + (number of headers +1, if present)
       + (number of trailers +1, if present)
```

This operator may appear only once in the input stream and has the following format:

NLINES = numeric literal
zero

Note: = or one of its synonyms may appear between the operator and the literal.

During a QUIP/RIT execution (see section 3.5.13.2, LOAD RIT operator), NLINES will override the BODYLINES in the RIT. If omitted, the output will be as follows:

On-Line - QUIP will set the BODYLINES field in the RIT to a high value to force the display mode of output so that a header will precede the first line of the output, and a trailer will follow the last line of output.

Batch - the BODYLINES field in the RIT will be used and headers and trailers will be printed on each page.

Example 1

```
NLINES 0
```

Comments: This example shows how the NLINES operator may be used to force the display mode of output.

Example 2

```
NLINES EQ 48
```

Comments: This example causes the body of the report to be brought to the center of the page.

3.5.3 Imperative Operators

LIST, DISPLAY, MARK and PRINT are called Imperative Operators since one of them must appear in every QUIP job which is not executing a stored query or RIT. These operators are used to generate output.

If the query does not have at least one of these operators but does have a retrieval or LIMIT operator, a print operator is internally generated as part of the following operator sequence.

COUNT FINAL PRINT COUNT

which is appended to the query. These added operators output the count of qualifying records for the query.

3.5.3.1 LIST Operator

The LIST operator gives the user the capability to list the contents of any field and/or group in the file and also enables him to produce a neat columnar report. It will generate as many columns and output lines as needed to list all of the specified fields; however, if all the fields will not fit on a single line of output, the fields will be sorted by set number and displayed accordingly. When more than one output line is generated for a record, all lines, except the first, will be five characters shorter (indented five characters) than the normal line; and the columnar arrangement of the output will be destroyed (see example 2).

Column headings are generated automatically by the LIST operator and will be preceded by three blank lines and followed by one. These column headings contain the output labels associated with the specified fields/groups by File Structuring (FS). If no output label was assigned to the field/group in the FFT, the field/group name will be used.

The fields/groups following the LIST operator may be edited or converted by a subroutine, but they may not be modified by partial field notation. In addition, the ALL modifier may precede any periodic field/group name to cause the listing of all subsets rather than just the flagged ones.

The format of the LIST operator is:

```
                 {fieldname}  [[EDIT]  "           ]
LIST  [ALL]      {groupname}  [[EDIT]  'edit mask' ]  . . .
                              [##                  ]
                              [#subtab#            ]
```

a. '' (double apostrophe) suppresses the FFT edit mask. (The word EDIT may precede the ".)

b. An edit mask, enclosed in apostrophes, may follow the field/group name to override the FFT-specified mask or to edit the field on output. The word EDIT may precede the edit mask.

c. ## (double pound sign) suppresses automatic table conversion.

d. An output subroutine, enclosed in pound signs, may follow the field/group to force table conversion.

Note: The size of the column is determined by the larger of the length of the field on output and the length of the output label plus two.

Example 1

```
LIST UIC UNAME MEQPT MEPSD
```

Comments: A columnar report will be produced, and the format of the output will appear as follows:

UNIT CODE	UNIT NAME	EQP. ID	EQP. POSSESSED
xxxxxx	xxxxxxxxxxxxxxxxx	xxxx	xxx
		xxxx	xxx
xxxxxx	xxxxxxxxxxxxxxxxx	xxxx	xxx
xxxxxx	xxxxxxxxxxxxxxxxx	xxxx	xxx
		xxxx	xxx
		xxxx	xxx

Example 2

```
LIST UIC UNAME COMDR PERS MECL LOC MEQPT
```

Comments: When the LIST operator contains more fields than can be contained in one output line, the columnar arrangement is destroyed and the title lines are repeated

for every record. In this case, the output would have the following format:

TERMINAL PROCESSING (TP)

UNIT CODE	UNIT NAME	COMMANDER	PERSNL	LOCATION
xxxxxx	xxxxxxxxxxx	xxxxxxxxxxxxxxxx	xxx	xxxxxxxx

MAJOR EQP. CLASS	MAJOR EQP. ID
xxx	xxxxx
xxx	xxxxx

UNIT CODE	UNIT NAME	COMMANDER	PERSNL	LOCATION
xxxxxx	xxxxxxxxxxx	xxxxxxxxxxxxxxxx	xxx	xxxxxxxx

MAJOR EQP. CLASS	MAJOR EQP. ID
xxx	xxxxx
xxx	xxxxx

UNIT CODE	UNIT NAME	COMMANDER	PERSNL	LOCATION
xxxxxx	xxxxxxxxxxx	xxxxxxxxxxxxxxxx	xxx	xxxxxxxx

MAJOR EQP. CLASS	MAJOR EQP. ID
xxx	xxxxx
xxx	xxxxx

Example 3

```
LIST ALL MECL ALL MEQPT ALL MEPSD.
```

Comments: During retrieval runs, the ALL modifier causes processing of all subsets rather than just the flagged ones.

Example 4

```
LIST SERV ## UNAME LOC PERS ''
```

Comments: The output subroutine (OCMDS) associated with the field SERV will be suppressed. In addition the field PERS will not be edited on output.

3.5.3.2 DISPLAY Operator

The DISPLAY operator allows the user to display literals and/or the value of file fields, sortkey entries, and the user-named result fields from a *COMPUTE* statement (see section 3.5.4.3, COMPUTE Operator). The combined length of the displayed information must not exceed the output line length. The format of the DISPLAY statement is:

```
                  ┌                                                    ┐
                  │ 'literal'                                          │
                  │ SORTKEY [m/n]                                      │
                  │                  ┌                          ┐      │
                  │                  │EDIT    ''                │      │
                  │                  │EDIT    'edit mask'       │      │
DISPLAY [nn]      │ [ALL] field      │#subtab#           [*mm]  │      │
                  │                  │##                        │      │
                  │                  └                          ┘      │
                  │[ALL] (field [nn] field [nn]...field [pp]) *mm       │
                  └                                                    ┘
```

Note: The above options may be entered separately or in combination with one another.

a. nn is a 1- or 2-digit number representing the number of blanks required before the next field or literal in the output line.

b. ALL is a modifier which allows for the processing of all periodic subsets rather than just the flagged ones.

c. m/n indicates partial field notation and specifies the portion of the sortkey to be displayed.

d. EDIT is a keyword which identifies an edit specification. It may be followed by double apostrophes ('') to suppress an FFT edit mask or by an edit mask enclosed in apostrophes to force an edit mask.

e. A subroutine expression can be used to modify a field or group name. A double pound sign (##) will supress automatic conversion, and a subroutine/table name enclosed in pound signs will force table conversion.

f. *mm is an operand which contains a 1- or 2-digit number representing the number of horizontal repetitions required for the preceding field or combination of fields. Note that the first character of this operand is an asterisk and must be coded as shown.

g. pp is a 1- or 2-digit number representing the number of blanks required between the horizontal occurrences of the field or combination of fields.

The values of the fields will immediately precede and/or follow the literals; hence, the user must indicate the number of spaces desired at the beginning of the line and between fields and/or literals. This is accomplished by specifying a 1- or 2-digit number (nn) which indicates the number of blanks before the following field or literal in the output line.

With periodic information, the user may specify that the requested subsets be displayed in a horizontal or vertical manner. In addition, he may request that only the flagged subsets be printed or that all subsets--flagged or unflagged--be printed. The following paragraphs discuss these capabilities.

Horizontal occurrences of a periodic field may be indicated by following the fieldname with a word (*mm) consisting of an asterisk as the first character and a 1- or

2-digit number which specifies the number of repetitions on the line. The line will be repeated, if necessary, with periodic information only, until the set is exhausted. If more than one field from the same periodic set is to have multiple horizontal occurrences, each field must be specified as part of one combination. A combination of fields is defined by enclosing in parentheses those fields to be repeated with their corresponding spacing parameters, e.g., (MEORC 3 MEORN 3 MEPSD) *4. All fields in a combination must be from the same periodic set. Without the presence of this word (*mm), there will be one occurrence of the field or combination per line (vertical display), and as many lines will be displayed as there are subsets to be processed.

If horizontal occurrences are specified for a field or a field combination, the normal spacing between occurrences in the output line is two spaces. The user can override this default spacing by coding after the fieldname, or after the last fieldname in a field combination, a 1- or 2-digit number (pp), which indicates the number of spaces required between occurrences. When this spacing override is used, the entire entry must be enclosed in parentheses, e.g., (MEORC 4) *3 or (MEORC 3 MEORN 3 MEPSD 4) *3.

The ALL modifier may be used with the DISPLAY operator to print all of the periodic subsets rather than just the flagged ones. When used with a combination, the ALL modifier produces identical results whether it is coded before the entire combination or before any fieldname within the combination. This specification is not necessary when printing a report directly from the data file since all subsets will be processed.

Example 1

```
DISPLAY 5 UNAME 3 PERS 7 MECL
```

Comments: The output line will be indented five characters. It will include the unit's name, followed by three spaces; the unit's personnel strength, followed by seven spaces; and the classes of equipment assigned to the unit, displayed vertically.

Example 2

```
DISPLAY MEQPT *4
```

Comments: A horizontal display of MEQPT has been requested. There will be four occurrences of MEQPT on one line, with two spaces (default spacing) between each occurrence, the first being left-justified on the output line. The line will be repeated until the set is exhausted.

Example 3

```
DISPLAY (MEQPT 5) *4
```

Comments: Same comments as above except that there will be five spaces between each occurrence.

Example 4

```
DISPLAY ALL (MEQPT 5) *4
```

Comments: Same comments as above except that all subsets--flagged and unflagged--will be displayed.

Example 5

```
DISPLAY ALL MEQPT
```

Comments: MEQPT will be displayed vertically. Each occurrence will be left-justified on the output line and all subsets will be displayed.

Example 6

```
DISPLAY 25 AVA
```

Comments: The COMPUTE result field identified by the literal AVA will be indented 25 spaces on the output line. This result field must have been previously defined by a COMPUTE operator. (See section 3.5.4.3, COMPUTE Operator.)

Example 7

```
DISPLAY SORTKEY 1/8
```

Comments: The first eight characters of the sort key will be displayed left-justified on the output line. A SORT statement must have preceded this specification.

Example 8

```
DISPLAY 5 (MEQPT 3 MEPSD 6) *5
```

Comments: A horizontal display of a field combination has been specified. There will be five occurrences of the combination on one output line, with six spaces between each occurrence. The first occurrence of MEQPT will be indented five characters, and there will be three spaces between each MEQPT and MEPSD. The line will be repeated until the set is exhausted. This example is only valid in batch QUIP where the output line length is 132 characters. It will not run with an output line length of 80 characters because the combined length of the displayed information cannot exceed the output line length.

Example 9

```
DISPLAY (PER1 2 ALL PER2) *3
```

Comment: The ALL modifier refers to each field in the combination regardless of where it is placed. Therefore, the above statement could have been written in either of the following forms:

a. DISPLAY ALL (PER1 2 PER2) *3

b. DISPLAY (ALL PER1 2 PER2) *3

3.5.3.3 MARK Operator

The MARK operator will cause a line of asterisks to be generated. It may be followed by a literal causing the literal to be embedded in the center of the line. This literal need not be enclosed in apostrophes unless it contains blanks, special characters, or reserved words. The format of the MARK operator is:

```
MARK  [literal]
```

Note: The literal must not exceed 69 characters in length.

Example 1

```
MARK
```

Comments: A row of asterisks will be output.

Example 2

```
MARK 'FINAL REPORT'
```

Comments: When it is output, the literal, FINAL REPORT, will be embedded in a row of asterisks.

3.5.3.4 PRINT Operator

The PRINT operator causes the sum or count work areas to be displayed. This operator is followed by one or more SUM or COUNT operators and, when encountered in the logic flow, causes these operands to be printed and reset to zero. The format of the PRINT operator is as follows:

```
          SUMn      ...  SUMn
PRINT     COUNTn         COUNTn
```

The PRINT operator outputs six lines: the first three are blank, the fourth line is a label line which identifies the specified SUM or COUNT operator and its associated work areas, the fifth is also blank, and the sixth contains the contents of the specified work areas.

Example

```
PRINT  SUM1 COUNT2
```

The PRINT operator may be followed by one or more SUM or COUNT operators. In this example both the SUM1 and COUNT2 work areas will be displayed and reset to zero.

3.5.4 Arithmetic Operators

3.5.4.1 SUM Operator

The SUM operator, which is followed by one or more numeric names of numeric fields, causes the value of the field to be added to the contents of the appropriate sum work area. This work area is two bytes longer than the field being summed unless the field name is followed by a 1- or 2-digit number (from 1 to 15) to identify the sum work area length required for the field. This user-specified length is optional for each field being summed and the value used applies only to the preceeding field. If more than one SUM operator are used during the run, the operator must be followed by a digit to identify each as a separate operator, i.e., SUM0...SUM9. The sum work areas are displayed by using the operator in a PRINT operation. These work areas are reset to zero every time they are printed.

The ALL modifier may precede any periodic fieldname to cause both flagged and unflagged subsets to be summed.

The SUM operator, with field operands and modifiers omitted, may be repeated in a different location in the program if the associated field is to be added to the original sum work area.

The SUM operator may also be used within the range of the FOR operator to form one- or two-dimensional arrays (see Section 3.5.7.1, FOR Operator).

The format of the SUM operator is described below:

SUMn [ALL] fieldname[m] --- [ALL] fieldname[m]

a. n is a digit from 0 to 9 which uniquely identifies the SUM operator. It is required when more than one SUM operator are used.

b. The ALL modifier causes processing of all subsets.

c. Only numeric fields (binary or decimal) may be summed.

d. m is a 1- or 2-digit number (from 1 to 15) used to assign a length to the SUM work area for the field preceeding the number. If this number is omitted, the default value for the length of the SUM work area is the length of the field being summed plus two.

Example 1

```
SUM1 PERS
```

Comments: When SUM1 is output, it will contain the total value of all authorized personnel.

Example 2

```
SUM5 MEPSD 7 MEORN MEORC
```

Comments: Three work areas are generated for this operator. The contents of each field is summmed in its corresponding work area. The length of the work area for the sum of MEPSD is seven bytes, whereas the length of the work areas for MEORN and MEORC is two bytes longer than the length of the corresponding field. When SUM5 is printed, the contents of all three work areas is printed and each work area is reset to zero.

3.5.4.2 COUNT Operator

The COUNT operator, which may be followed by one or more alpha or numeric fieldnames, is used to count the occurrence of records which contain the specified fields. The record is not counted when the field specified is alphabetic and its value is blanks or when the field specified is numeric and its value is zero or blanks. If no fieldnames follow the first occurrence of a COUNT operator, the fixed set will be counted by default. If more than one COUNT operator is used during the run, the operator must be followed by a digit to identify each as a separate operator; i.e., COUNT0...COUNT9. The count work areas, which are six bytes long are displayed by using the operator in a PRINT operation. These work areas will be reset to zero every time they are printed.

The ALL modifier may precede any periodic fieldname to cause both flagged and unflagged subsets to be counted.

The COUNT operator, with field operands and modifiers omitted, may be repeated in a different location in the program if the associated field is to be added to the original count work area.

The COUNT operator may also be used within the range of the FOR operator to form one- or two-dimensional arrays (see section 3.5.7.1, FOR Operator).

The format of the COUNT operator is described below:

COUNTn [ALL] fieldname .. [ALL] fieldname

a. n is a digit from 0 to 9 which uniquely identifies the COUNT operator. It is required when more than one COUNT operator is used.

b. The ALL modifier causes processing of all subsets.

c. Both alpha and numeric fields may be counted.

Example 1

```
COUNT
```

Comments: The COUNT operator may appear by itself to count the fixed set.

Example 2

```
COUNT1 ALL MEQPT
```

Comments: Generates one work area which counts all of the specified subsets.

3.5.4.3 COMPUTE Operator

The COMPUTE operator is used to specify various arithmetic and statistical operations. The user may employ this operator to perform the arithmetic operations of

addition, subtraction, multiplication, and division with file fields, predefined literals, and self-defining literals; in addition, he may use this operator with file fields to calculate the statistical functions of average, percent, variance, maximum and minimum. During retrieval runs, this operator is used only against the fixed set and flagged subset fields unless the ALL modifier is used. If the ALL modifier is used, all subsets are processed for the entire statement. The COMPUTE operator is associated with the three format types shown below; each format type is determined by the function being performed:

When performing arithmetic operations, the following format is used:

COMPUTE user-name [xx] EQ { [ALL] fieldname / literal } operation { [ALL] fieldname / literal } operation... { [ALL] fieldname / literal }

a. The user-name is the name assigned by the user to be applied to the result. This name must not be a fieldname or a reserved word and must not have been previously defined.

b. xx is a 1- or 2-digit number (from 1 to 32) used to assign an output length to the literal defined by the user-name.

c. EQ or one of its synonyms must be the relational operator used.

d. The ALL modifier allows for the processing of all periodic subsets rather than just the flagged ones.

e. The fieldname must be numeric, i.e., binary or decimal, and must be from the fixed set or one periodic set.

f. The literal may be the user-name defined by a previous COMPUTE statement; or it may be a self-defining numeric literal which is an integer less

than 16 digits long (a self-defining numeric literal may be signed or unsigned).

g. The operation must be identified by one of the following symbols:

+ (for addition)
- (for subtraction)
* (for multiplication)
/ (for division)

The operation symbol must be preceded and followed by one or more blanks.

When performing statistical functions, the following format is used:

COMPUTE user-name EQ function fieldname(s)

a. The user-name is the name assigned by the user to be applied to the result. This name must not be a fieldname or a reserved word and must not have been previously defined.

b. EQ or one of its synonyms must be the relational operator used.

c. The function is defined as AVERAGE, PERCENT (%), VARIANCE, MAXIMUM or MINIMUM.

d. The fieldname(s) must be numeric; i.e., binary or decimal, and must be from the fixed set or one periodic set.

When performing the action of a previously defined COMPUTE statement, the following format is used:

COMPUTE user-name

The user-name is the name assigned by the user to be applied to the result. It must have been defined previously by another COMPUTE statement.

The results (identified by the user-name) of any COMPUTE operation are output and reset to zero in a DISPLAY statement. This result area must be defined in a COMPUTE statement before it is displayed. When periodic fields are being processed, the result literal will contain the results after computations on the last subset.

3.5.4.3.1 Arithmetic Operations

A COMPUTE operator containing one or more operation symbols (+,-,*,/) defines an arithmetic expression. An expression defined in this manner will be computed serially from left to right and may not contain parentheses. If file fields have been referenced, they must be from the fixed set or from the same periodic set. The user may specify an external length of up to 32 bytes for the result literal defined by this expression. If a length is not specified by the user, a default value (maximum of 32) is computed according to the following rules:

a. Proceeding serially left to right, the length of a product is the sum of the lengths of the fields or literals.

b. The length of a quotient is the length of the dividend.

c. The length of the result of a string of consecutive add and/or subtract operations is the length of the longest field or literal in the string plus 1 for every 10 consecutive operations or fraction thereof.

d. The length of a result literal which is designed to maintain a running total (an expression where the result literal is on both sides of the EQUAL, e.g.,

```
COMPUTE LITA EQUAL LITA + FLDA
```

such that LITA contains the total of all FLDA) is twice the length of the field obtained by the procedures in (a), (b), and (c).

The current result value will be kept in a 16-byte work area, with overflow causing the field to be filled with asterisks when it is output.

3.5.4.3.2 Statistical Operations

3.5.4.3.2.1 AVERAGE Function

The average value of a numeric field may be found by using the AVERAGE function. The value of the field (or fields if in periodic set) will be added into a work area and the field count bumped by one (more) every time this operator is encountered in the logic flow. The actual calculation of the average will occur at the time the value is to be output. The work areas will then be reset to zero.

The current sum will be kept in a fullword, with overflow causing the field to be filled with asterisks when it is output (maximum sum 2,147,483,647). The current count of fields will be kept in a halfword which limits it to less than 32,767.

The format of the average when it is output will be:

```
                AAAA.A    (NNNNN)
where:          AAAA.A  = the average (calculated to tenths)

                NNNNN   = the number of fields used in the
                          average
```

The data field being averaged is assumed to be a whole number. The number of characters in an average equals the field output length plus a decimal point, plus a character for tenths.

3.5.4.3.2.2 PERCENT Function

The percent that the sum of one numeric field is of the sum of another may be calculated using the PERCENT function. The value of the two fields will be summed in two work areas every time this operator is encountered in the logic flow. The actual calculation of the percent will occur at the time

the value is output. The work areas will then be reset to zero.

The sums will be kept in fullword work areas, with overflow causing the field to be filled with asterisks when it is output. The format of the percent when it is output will be:

PP.P%

The two fields must be from the same periodic set if they are not fixed set fields.

3.5.4.3.2.3 VARIANCE Function

The variance of a numeric field may be calculated by the use of the VARIANCE function. The formula used to compute the variance is:

$$\text{VARIANCE} = \frac{\sum xi^2}{N} - \frac{\sum xi}{N}^2$$

The value of the field and the value of the field squared will be summed, and the subsets will be counted every time this operator is encountered in the logic flow. The actual calculation of the variance will occur at the time the value is output. The work areas will then be reset to zero.

The sums will be kept in fullword work areas, with overflow causing the field to be filled with asterisks when it is output. The count will be kept in a halfword work area. This limits the number of subsets allowed to less than 32,767. The format of the variance when it is output will be:

VV.VV (NNNNN)

where: VV.VV = the variance (calculated to hundredths)

NNNNN = number of subsets involved in calculation

3.5.4.3.2.4 MAXIMUM Function

The maximum value of a numeric field may be found by using the MAXIMUM function. The value for the first occurrence of the field is saved in a work area. The value for each following occurrence is compared to the value in the work area. If the value of the field is greater than that of the work area, that higher value replaces the current entry in the work area. The length of the result area is equal to the length of the field. A field which contains a zero value is not considered when computing the maximum value. The result of a MAXIMUM function may be used in an arithmetic type compute statement. If no maximum value has actually been computed, the value of the MAXIMUM result is zero for purposes of the arithmetic compute operation.

3.5.4.3.2.5 MINIMUM Function

The minimum value of a numeric field may be found by using the MINIMUM function. The value for the first occurrence of the field is saved in a work area. The value for each following occurrence is compared to the value in the work area. If the value of the field is less than that of the work area, that lower value replaces the current entry in the work area. The length of the result area is equal to the length of the field. A field which contains a zero value is not considered when computing the minimum value. The result of a MINIMUM function may be used in an arithmetic type compute statement. If no minimum value has actually been computed, the value of the MINIMUM result is zero for purposes of the arithmetic compute operation.

3.5.4.3.2.6 Examples of Statistical Operators

Example 1

```
COMPUTE X 10 EQ A + PERS * 2
```

Comments: A result field labeled X has been defined to have a length of 10. It holds the results of an arithmetic expression containing a previously defined COMPUTE result

area (A), a file field (PERS), and a self-defining numeric literal (2).

Example 2

```
COMPUTE X EQ A + PERS * 2
```

Comments: Same comments as above except that the length of X has been assigned a default value.

Example 3

```
COMPUTE AVA = AVERAGE PERS
```

Comments: A result field labeled AVA has been defined to contain the average of PERS.

Example 4

```
COMPUTE PRCNT EQ % MEORN OF MEPSD
```

Comments: A result field labeled PRCNT has been defined to contain the percent MEORN is of MEPSD.

Example 5

```
COMPUTE VAR EQ VARIANCE FLDZ.
```

Comments: A result field labeled VAR has been defined to contain the variance of FLDZ.

Example 6

```
COMPUTE LARG EQ MAXIMUM PERS
```

Comments: A result field labeled LARG has been defined to contain the maximum value of the field PERS.

3.5.5 Conditional Operators

Conditional operators permit the user to execute output operators only when a certain condition has been satisfied or an end-of-file has been sensed.

3.5.5.1 Output Type IF Operator

The output type IF operator is followed by one conditional clause and one or more output operators. It causes the output operator(s) within the statement to be executed only when the condition is found to be true. This operator must be terminated with a period and follows the format shown below:

```
                 { fieldname                }
IF               { groupname [#subtab#]     } relational          output
                 { SORTKEY [m/n]            } operator  [literal]  operator(s).
```

Conditional Clause

a. The field or group specified must be from the fixed set unless the relational operator is ABSENT, in which case a field from a periodic set or the variable set must be referenced. The field or group specified may be modified by a subroutine if the relational operator is not ABSENT or CONTAINS. The field or group specified must be alphabetic mode when the CONTAINS relational operator is used. If the SORT operator (see Section 3.5.11, SORT Operator) has been used, the SORTKEY may be specified to reference fixed and periodic fields. Partial field notation may be used to address portions of the SORTKEY.

b. The valid relational operators are: EQ, LT, LE, GT, GE, NE, BT, CHANGES, CONTAINS, ABSENT and any synonym associated with these operators (see Section 3.12, Relational Operators).

c. The literal is a required entry with all relational operators except CHANGES and ABSENT. For all relational operators except CONTAINS, it will be

converted automatically to its coded form if an input subroutine or table was specified for the field in the File Format Table (FFT). Only a single literal is allowed. Self-defining literals which are reserved words or contain special characters must be enclosed in quotations marks to be interpreted as a valid literal.

d. The output operator(s) specifies the action to occur if the condition is met. The following operators may be used: COMPUTE, COUNT, DISPLAY, EXTRACT, EJECT, LIST, MARK, PRINT, SPACE, and SUM.

e. Whereas the normal conditional clause is satisfied based on the contents of certain fields in the record, the ABSENT operator provides the capability to test the absence of a specific periodic or variable set. It causes the referenced set, identified by a field or group name from that set, to be examined. If no subset exists for the record, the condition is true.

Note: The IF statement will be examined by the system to determine whether or not it contains any output operators. If it does, it will be considered to be of the output type and handled accordingly.

Example 1

```
IF PERS GT Ø COMPUTE AVA EQ AVERAGE PERS.
```

Comments: Arithmetic and statistical computations may be conditioned.

Example 2

```
IF SERV CHANGES PRINT SUM1 EJECT.
```

Comments: The PRINT and the EJECT operators will be executed only when SERV changes.

Example 3

```
IF MEQPT ABSENT LIST UIC.
```

The LIST operator will be executed for only those records which do not have any subsets for periodic set one.

Example 4

IF COMMENT CONTAINS 'SOUTH' LIST UIC.

The UIC will be printed for those records where the character pattern SOUTH occurs anywhere in the variable length field COMMENT.

3.5.5.1.1 INITIAL Modifier

The CHANGE logic used in QUIP does not sense a change condition on the first record. Consequently, when the IF CHANGE operator is first encountered in the input stream, control is passed to the end of the statement; and the statement is not executed. If the user desires to execute all or part of this IF statement, he may use the INITIAL modifier. The INITIAL modifier allows part (or all) of the range of an output type IF operator, with the CHANGES relational operator, to be executed on the first record. The first time the IF is encountered, control is passed to the INITIAL modifier instead of to the end of the statement.

The format of the output type IF operator used with the INITIAL modifier is like the one described above, except that the word INITIAL may appear anywhere after the CHANGES relational operator and before the last output operator in the statement. (Recall that the literal is not a valid entry for the CHANGES relational operator and, therefore, would not appear in this statement.)

Example

```
IF SERV CHANGES
PRINT SUM3 COUNT6
INITIAL LIST SERV LOC.
```

Comments: The SUM and COUNT work areas are not printed on the first record, but the fields, SERV and LOC, are listed.

3.5.5.2 FINAL Operator

The FINAL operator conditions all of the operators following it so that they will be executed only after an end-of-file condition has been sensed on the data base. This capability permits the printing of "final" totals and summaries after all of the data base processing has been completed. The FINAL operator is not allowed within the range of an EXTRACT operator. Operators which reference data; i.e., SUM, COUNT, LIST, may not follow the FINAL operator. A statement terminator should logically precede the FINAL operator, but one will be inserted by QUIP if it is not present.

The format of the FINAL operator is:

```
                  { COMPUTE }
                  { DISPLAY }
                  { EJECT   }
FINAL             { MARK    }
                  { PRINT   }
                  { SPACE   }
```

The COMPUTE operator may follow the FINAL operator, but it must be an arithmetic type operator and must not reference any file fields.

3.5.6 Forms Control Operators

3.5.6.1 SPACE Operator

The SPACE operator may be used to generate spaces (blank lines) in the output. It may be followed by a numeric literal to indicate the number of lines to space. If a numeric literal is not present, one space will be generated. If the number of lines to space is greater than the number of lines left on the current page when the output mode is paging, the operation will result in an EJECT, and the remaining spaces will not be spaced on the new page. The SPACE operator appears among the output operators and executes in the sequence it appears. It may also appear within the range of an output 'IF' condition and the EXTRACT operator. The SPACE operator should not appear within the range of HEADERS or TRAILERS. Spacing between HEADERS or

TRAILERS may be accomplished by inserting a HEADER/TRAILER with a blank literal. The format of the SPACE operator is as follows:

SPACE [numeric literal]

Example 1

SPACE

Comments: One blank line will be generated.

Example 2

SPACE 3

Comments: Three blank lines will be generated.

3.5.6.2 EJECT Operator

The EJECT operator will cause the current page to be terminated and the next page to be initiated when the paging mode of output is used. In the display mode of output, this operator will cause the generation of extra header lines and, consequently, should not be used.

The format of the EJECT operator is:

EJECT

3.5.7 Matrix Generation Operators

Operators in this category permit the user to generate a one- or two-dimensional matrix.

3.5.7.1 FOR Operator

The FOR operator is used to generate a one-dimensional matrix. It is followed by a conditional clause and one or more SUM and/or COUNT operators with an expanded format for use with the FOR operator. The range of the FOR operator is terminated by the next operator other than a SUM or COUNT or by a statement terminator. The format of the FOR operator with the SUM and COUNT operators is:

```
FOR  fieldname  ##        relational
               [#subtab#]operator   literal(s)

 {SUMn[ALL]fieldname[m]---[ALL]fieldname[m]}    [HTOTAL]
 {COUNTn[ALL]fieldname---[ALL]fieldname    }

 ---  {SUMn[ALL]field[m]---[ALL]fieldname[m]}    [HTOTAL]
      {COUNTn[ALL]fieldname---[m]fieldname  }
```

a. The fieldname may identify a fixed or periodic field and may be modified by an input subroutine. If the field has automatic table conversion, the user may suppress the conversion with a double pound sign (##).

b. The valid relational operators are: EQ, LT, LE, GT, GE, NE and any synonyms associated with these operators. The NOT modifier may precede any of these relational operators.

c. Only self-defining literals are allowed. Literals which are reserved words or contain special characters (except for the special use of a single asterisk discussed below) must be enclosed in quotation marks to be interpreted as a valid literals. The literals establish one of the output title lines and determine the columns of the matrix. QUIP program restrictions provide that a maximum of 15 literals may be defined. However, the maximum for a particular conditional clause may be less than 15, depending on a number of factors: the output line width, the size of the literals in the output title line, the size of the SUM/COUNT work areas, and a request for horizontal totals. The leftmost column is indented two positions and consists of the name of the field being counted or summed. The remaining available space is allocated evenly to the matrix columns, the number of columns being determined by the literals and the HTOTAL, if present, which are spread across the page or screen using all available positions.

   In a COUNT matrix, the width of each column is six bytes, the size of the COUNT work area. In a SUM

matrix, the width of each column is either the user-specified size for the SUM work area or the default size equal to the length of the field being summed plus two.

Each literal is associated with an accumulation result area in which the SUM or COUNT totals are maintained. An asterisk may be used as the last literal in the operand string to provide a work area to maintain a SUM or COUNT total for those data fields that were not summed or counted for any of the specified literals. (Subroutine or table conversion does not apply to this "asterisk" result area.)

d. One or more SUM or COUNT operators must follow the conditional clause. These SUM or COUNT operators identify the field or fields to be summed or counted.

e. The HTOTAL modifier (see section 3.5.7.2) generates a horizontal total on output.

Output from this operator is displayed by the execution of a PRINT operator referencing the specified SUM or COUNT work areas. The output consists of two title lines and a matrix. The first title line will be a skeleton of the FOR operator specified in the input. This skeleton contains the FOR operator and the fieldname and relational operator which follow it and provides a key to interpreting the data of the matrix. The second title line is composed of the name of the SUM or COUNT operator and the input FOR literals. The matrix contains the SUM or COUNT work areas listed below the corresponding literals.

Specifying SUMs or COUNTs with multiple fieldnames causes the creation of a one-dimensional matrix (horizontal) for each fieldname present. Each matrix, when printed by a PRINT operator, will be printed in the order specified by the SUM or COUNT operator.

The use of a relational operator other than EQ will normally provide matrix values reflecting a range of field values. A value is summed or counted no more than once for

a matrix row. Thus, care must be taken when specifying the order of multiple literals if EQ is not used. For example, if LE is to be used, literals must be in ascending sequence to obtain the desired columnar values.

The use of a dollar sign ($) in a literal not enclosed in single quotes indicates a universal match character (see section 3.7, Universal Match Character) and causes that character position to be ignored when compared to its corresponding character in the data base field. Consequently, care should be taken when using this specification since the first qualifying literal stops the search (see example 3 on the next page).

Example 1

```
FOR SERV EQ ARMY NAVY SUM1 PERS.
```

Comments: Generates a one-dimensional matrix containing the total number of authorized personnel for the ARMY and the NAVY. The format of the matrix when it is output is:

```
FOR SERV EQ
SUM(1)    PERS   ARMY   NAVY
                 XX     XX
```

Example 2

```
FOR FLDX LT 30 50 100 COUNT FLDY.
```

Comments: When the matrix is output, the values listed under the column labeled 50 will be greater than 30 but less than 50. Similarly, the values listed under the column labeled 100 will be greater than 50 but less than 100.

Example 3

```
FOR FLDA EQUALS A$ AB$ AA$ SUM FLDB.
```

Comments: This statement never has the opportunity to add to the SUM work area for the AB$ and AA$ values since they have already qualified by the A$ value. If the statement is changed to read FOR FLDA EQUALS AA$ AB$ A$ SUM FLDB., the A$

accumulation bucket will include all values beginning with A except those with a second character of A or B.

Example 4

```
FOR FLDB EQ LIT1 LIT2 * SUM FLDX 6 FLDY 7.
```

Comments: Two 1-dimensional matrixes will be generated, one for FLDX and one for FLDY. The sum work areas for FLDX and FLDY will be six bytes and seven bytes long respectively. The asterisk in the literal string will cause a sum work area to be generated which will accumulate the total of the sum field for those records where FLDB does not equal LIT1 or LIT2.

3.5.7.1.1 FOR Operator With the AND Logical Connector

The AND logical connector is used with the FOR operator to add a second dimension to the FOR matrix. The AND connector is placed after the FOR conditional clause which is followed by another conditional clause and one or more SUM and/or COUNT operators with an expanded format for use with the FOR operator with AND logical connector. The fields referenced in each clause must be from the same set or one of the fields must be from the fixed set. The format of the FOR operator with AND logical connector with the SUM and COUNT operators is:

```
                  [##       ]relational
FOR fieldname     [#subtab# ]operator  literal(s)
                  [##       ]

                  [##       ]relational
AND fieldname     [#subtab# ]operator literal(s)
                  [         ]

  {SUMm [ALL]fieldname --- [ALL]fieldname[m][VLONG]}
                                                    [HTOTAL]
  {COUNTn [ALL]fieldname --- [ALL]fieldname        }

---{SUMn [ALL]fieldname[m] --- [ALL]fieldname[m][VLONG]}
                                                       [HTOTAL]
   {COUNTn [ALL]fieldname --- [ALL]fieldname           }
```

a. The fieldnames may identify fixed or periodic field. However, if the fields are both periodic, they must be from the same set. If one field is from the fixed set and one is periodic the periodic must be referenced in the FOR and the fixed in the AND connector. Either or both fields may be modified by an input subroutine or table. If the field has automatic conversion, the conversion may be suppressed with a double pound sign(##).

b. The valid relational operators are EQ, LT, LE, GT, GE, NE and any of their synonyms. The NOT modifier may precede the relational operator.

c. Only self-defining literals are allowed. Literals which are reserved words or contain special characters (except for the special use of a single asterisk discussed below) must be enclosed in quotation marks to be interpreted as valid literals.

The output consists of two title lines and a two-dimensional matrix with the FOR literals forming the horizontal axis and the AND literals forming the vertical axis. A maximum of 15 literals may be specified with the FOR conditional clause and a maximum of 99 literals may be specified with the AND conditional clause. However, the total number of SUM or COUNT work areas associated with any FOR/AND expression may not exceed 1000. Each literal in the FOR conditional clause is associated with one or more accumulation result areas in which the SUM or COUNT totals are maintained. The number of result areas associated with each FOR literal is equal to the number of literals in the AND conditional clause. The elements of the matrix are SUM or COUNT work areas corresponding to the particular FOR and AND conditions which are satisfied along with the vertical totals and the optional horizontal totals.

The maximum number of literals that may be specified in a particular FOR conditional clause may in reality be less than 15, depending on a

number of factors: the output line width, the size of the literals in the output title line, the size of the SUM/COUNT work areas, a request for horizontal totals, and a request for expanded vertical totals with the VLONG operand.

The leftmost column consists of the name of the field being summed or counted and the list of literals from the AND conditional clause. The remaining available space is allocated evenly to the matrix columns, the number of columns being determined by the literals in the FOR conditional clause and the HTOTAL, if present, which are spread across the page or screen using all available positions.

In a COUNT matrix, the width of each column is six bytes, the size of the COUNT work area. In a SUM matrix, the width of each column is either the user-specified size for the SUM work area or the default size equal to the length of the field being summed plus two.

An asterisk may be used as the last literal in either or both conditional clauses to provide a work area to maintain a SUM or COUNT total for those data fields that were not summed or counted for any of the specified literals. (Subroutine or table conversion does not apply to this "asterisk" result area.)

d. One or more SUM or COUNT operators must follow the AND conditional clause. The SUM or COUNT operators identify the field or fields to be summed or counted. The field or fields summed must be from the same set as referenced in the AND conditional clause.

e. The VLONG operand is valid only for the SUM operator when used with the FOR operator with the AND logical connector. Its use increases the size of the vertical total result area by two bytes, i.e., length of SUM work area plus 2. It has no effect on the actual computation of the vertical

totals as they are always generated for each matrix. The absence of the VLONG operand indicates that the sizes of the vertical total result areas are to be the same as the sizes of the SUM result areas.

f. The HTOTAL operand (see section 3.5.7.2) generates the horizontal totals for the matrix.

A SUM or COUNT operator which specifies multiple fieldnames causes the creation of a two-dimensional matrix for each field. Each matrix is output in the same order as specified by the SUM or COUNT operator. The matrix(es) generated by the FOR operator with AND logical connector is displayed by means of a PRINT operator which referenced the name of the SUM or COUNT operator. The output for each matrix consists of two title lines and the work areas, including vertical and horizontal totals, if requested. The first title line is a skeleton of the FOR/AND statement consisting of the FOR operator and the AND connector with their corresponding conditional clauses less the literals. The second title line is composed of the name of the SUM or COUNT operator and the literals from the FOR conditional clause. The matrix contains the list of literals from the AND conditional clause in the leftmost column with each SUM or COUNT work area in the column corresponding to the particular FOR literal and in the row corresponding to the particular AND literal. As the bottom row of the matrix, the totals for each column are provided. If HTOTAL was specified, the horizontal totals form the rightmost column of the matrix.

Example

```
FOR SERV EQ ARMY NAVY USAF
AND LOC EQ PARIS METZ TOULON
COUNT PERS
```

Comments: Generates a two-dimensional matrix with the following output format:

```
FOR SERV EQ                AND LOC EQ
COUNT   PERS          ARMY NAVY USAF
        PARIS          X     X    X
```

METZ	X	X	X
TOULON	X	X	X
TOTAL	X	X	X

3.5.7.2 HTOTAL Modifier

The HTOTAL modifier may be used with a SUM or COUNT operator within the range of a FOR operator to cause creation of a horizontal total on output. The following example illustrates the use of this modifier.

Example

```
FOR MECL EQ A$ H$ *
AND LOC EQ CANNES MELUN VERDUN
SUM1 MEPSD HTOTAL.
```

Comments: The output of this example would appear as shown below:

FOR MECL EQ		AND LOC EQ			
SUM(1)	MEPSD	A$	H$	*	TOTAL
	CANNES	X	X	X	X
	MELUN	X	X	X	X
	VERDUN	X	X	X	X
	TOTAL	X	X	X	X

3.5.8 Interfile Output

Interfile Output (IFO) permits data from more than one file to be combined during the output process. The primary file contains data elements which identify records in referenced secondary files. Those records are retrieved and processed against secondary file logical specifications, after which primary file processing is resumed. The primary file is that file defined by the FILE operator. Secondary files are those files referenced by the EXTRACT operator described below. Up to nine secondary files may be referenced by the EXTRACT operator. All secondary files must be ISAM. The symbolic parameters, ISAM1 and ISAM2, are provided for two of the secondary files. If more than two secondary files are required, additional DD statements with

the names DATAFIL3 through DATAFIL9 are required, one for each additional secondary file.

When a secondary file is a non-NIPS file, the EXTRACT statement will also reference the FFT which describes the non-NIPS file. The symbolic parameters FFT1 and FFT2 are provided for two of the non-NIPS secondary files. If more than two non-NIPS secondary files are required, additional DD statements with the names FFT3 through FFT9 are required, one for each additional non-NIPS secondary file. The suffix number on the FFTn DD statement must match the suffix number on the DATAFILn DD statement.

3.5.8.1 EXTRACT Operator

The EXTRACT operator is used to initiate the IFO process. Its presence in the input stream signals the interruption of primary file processing and the retrieval from a secondary file of one or more data records, the selection of which is based on the contents of the pointer field used in the EXTRACT statement. These retrieved records are then immediately processed against the logical specifications coded for the secondary file and which follow the EXTRACT statement in the input stream. If the length of the field used as the pointer is equal to the length of the record key of the secondary file, at most one record is retrieved and its record key matches the pointer. If the field used as the pointer has length greater than the length of the record key of the secondary file, the pointer is truncated before retrieval and at most one record is retrieved with its record key matching the truncated field. If the length of the pointer field is less than the length of the secondary file record key, all secondary file records are retrieved where the high-order portion of the record key matches the pointer, the length of the high-order portion used being determined by the length of the pointer.

The syntax of the EXTRACT operator is:

```
                                                    [##
                                                    [#subname#
                                                    [m/n
EXTRACT [ { [ALL] IFOGRPx                           ]
          { [ALL] fieldname                         ]   filename
          { SORTKEY [m/n]                           ]
          { literal name                            ]
          {
          { [FFT fftname]
```

a. The ALL modifier allows for the processing of all periodic subsets rather than just the flagged ones.

b. IFOGRPx is a group name in the primary file. The first field in the group contains the secondary file name or an identifier which is converted to the secondary file name by a subroutine specified for the field in the FFT. The second field in the group contains the actual pointer to the secondary file records, or it contains a field which is converted to the pointer by a subroutine specified in the FFT. Any other fields in the group are ignored by the EXTRACT operator. Subroutine or table conversion cannot be negated or overridden, nor can it be specified dynamically on the EXTRACT statement.

c. fieldname is the name of the field in the primary file used as the pointer to the secondary file records. It may have a subroutine specified in the FFT, in which case the field is converted by the subroutine before the retrieval is made of secondary file records, unless the double pound sign is used to negate conversion. A binary field is converted to decimal and its output length is used by the EXTRACT operator to retrieve secondary file records. The field must not be a variable field.

d. If the SORT operator is used or the primary file is a RASP answer set, SORTKEY may be specified to reference the SORTKEY or any portion of it.

e. Literal name is the name of a previously defined COMPUTE result.

f. Subname is the name of a user-written subroutine or table which is to convert the field before retrieval of the secondary file records.

g. m/n indicates partial field notation and specifies the portion of the field or sortkey to be used as the pointer. It is not used with IFOGRPx, a compute result, or a field which is converted by a subroutine specified dynamically in the source statement. Its use with a field for which an FFT subroutine is specified negates the application of the subroutine.

h. filename is the name of the secondary file from which records with keys identical to the pointer are retrieved and then processed with the logical specifications for that file, which follow the EXTRACT operator in the input stream. If the file name is a qualified data set name, the field must be enclosed in apostrophes.

i. FFT indicates that the secondary file is non-NIPS and fftname is the name of the FFT which describes the format of the secondary non-NIPS file.

The logical specifications that apply to the records retrieved by an EXTRACT operator are coded immediately following the EXTRACT statement in the input stream. These specifications are referred to as the "range" of the operator. The EXTRACT operator without operands is used to terminate the range of a previous EXTRACT operator and cause the return to primary file processing. Alternately, the range of an EXTRACT operator is terminated by another EXTRACT operator or the end of the query.

If there is no record in the secondary file that matches the record key specified on the EXTRACT statement, processing of the current primary file record continues following the range of the EXTRACT operator.

If more than one record is retrieved by the EXTRACT operator, the entire range of the EXTRACT statement is processed for each secondary file record retrieved, after which processing of the primary file record is resumed following the range of the EXTRACT operator.

When the pointer is a periodic field or group, the EXTRACT operation and its associated secondary file processing are performed for each subset (flagged or ALL). After the set has been exhausted, processing of the primary file record is resumed following the range of the EXTRACT operator.

When an EXTRACT operator is encountered as an output operator within an output type IF statement, a terminating period is required after the file name to define the range of the IF operator. A period is internally supplied if it is missing. The conditional clause then applies to the entire range of the EXTRACT operator because, when the condition is not true, the EXTRACT operation is not performed and processing continues following the range of the EXTRACT operator. This allows output type IF operators to be included in the specifications for secondary file processing without ambiguity as to where the previous IF statement terminated.

The following operators are not allowed within the range of an EXTRACT operator: Retrieval Type IF, SORT, LIMIT, FIND, FINAL, FUNCTION, LOAD RIT. Reference to the sort key is also not allowed since only the primary file may be a RASP answer set. Reference to FUNCTION work areas is also invalid.

When the operand specified as the pointer on the EXTRACT statement is IFOGRPx, the EXTRACT operation is performed only for those records (subsets) where the filename or converted file identifier found in the first field of the IFOGRPx matches the file name specified on the EXTRACT statement.

3.5.9 Retrieval Operators

3.5.9.1 Retrieval Type IF Operator

The retrieval type IF operator provides the logical criteria for data selection. Only one such operator is permitted per query, and it must be terminated with a period. The retrieval type IF statement may not be used if a RASP Answer Set is used as input to QUIP. This operator is not allowed within the range of an EXTRACT operator. If this operator refers to an index field, an indexed retrieval will be attempted. How Index Processing interprets the retrieval logic and the rules by which it determines if indexing information can be utilized are discussed in section 3.5.9.3, Retrieval Operators and Secondary Indexing.

If the input data file is indexed and Secondary Index processing has not been bypassed by the user, the data selection logic associated with this operator will be analyzed to determine if an indexed retrieval can be performed. Secondary Index processing can be bypassed by using the INDEX=NO option in the PARM field of the EXEC statement for a query run in the background environment or by including the keyword INDEXNO as the first word of an online query.

The statement formed by the retrieval type IF operator consists of one or more clauses linked by the logical connectors AND or OR to form logical conditions. All clauses joined by AND must be satisfied for the logical block to qualify. Clauses joined by AND are evaluated together; then the OR logic is performed on the results. To modify this order, parentheses may be used. The clauses within parentheses are evaluated, and where nested parentheses are used, the expression contained in the most deeply nested parentheses (that is the innermost pair of parentheses) is evaluated first. Successively higher levels are evaluated until the truth factor for the entire statement has been determined. The format is:

```
IF [ANY] { fieldname  } [m/n][ #subtab# ][NOT] { geo. operator } literal(s)
         { groupname  }                        { rel. operator }
```

Conditional Clause

```
... { AND } Conditional Clause .
    { OR  }
```

a. The ANY modifier is used to modify the conditional logic so that successive terms against the same periodic set are not evaluated within one subset at a time.

b. The field or group name identifies the character string in the record to be tested against the data value. The field/group may be modified by partial field notation and/or a subroutine expression.

c. m/n indicates partial field notation and specifies the portion of the file field or group character string to be used in the compare. This expression may not be used with the geographic relational operators (CIR and OVP) or binary and coordinate mode fields. The use of partial field notation in a clause, will prohibit Secondary Indexing for that clause.

d. A subroutine expression, identified by a double pound sign (##) or the name of a subroutine enclosed in pound signs, may be used to modify a field or group. The first suppresses automatic table conversion, and the latter forces table conversion. Subroutine expression may not be used with geographic relational operators nor with the relational operators ABSENT and CONTAINS.

e. NOT reverses the truth value of the relational operator.

f. The valid relational operators are EQ, GT, GE, LT, LE, BT, NE and any of their synonyms (see Section 3.12, Relational Operators). The valid geographic operators are OVP, CIR and any of their synonyms (see section 3.13, Geographic Operators).

g. Additionally, there are two special purpose relational operators, CONTAINS and ABSENT. CONTAINS is a relational operator used to specify a pattern for which fixed-length alpha fields(with or without partial field notation), variable length fields, and variable sets may be searched. The pattern will be recognized if it is anywhere within a field or variable set. ABSENT is a relational operator used to specify the absence of a periodic or variable set. The operator must be used in conjunction with field or group names of a periodic or variable set. The name serves as indication of the particular set being referenced. No data value can be specified for an ABSENT operator.

The ABSENT conditional clause causes qualification at the record level, therefore, referenced subsets in qualifying records will not be flagged. The ALL modifier must precede all field/group names of the referenced set to cause the listing or display of values from the unflagged subsets. Further, any attempt to sort records on periodic fields from the referenced set would cause those qualifying answer entry records to be eliminated from the answer set.

h. The retrieval literals will be converted to their coded form if an input subroutine or table was specified for the field in the File Format Table (FFT) and the relational operator is not CONTAINS or ABSENT. The literals are padded with blanks to the length of the field or group prior to making the comparison. The number and type of literals required vary with the geographic or relational operator chosen. Basically they consist of self-defining and indirect address literals. Self defining literals which contain special characters or are reserved words must be enclosed in quotation marks to be interpreted as a valid retrieval literal. Special characters which are encountered in literals which are not enclosed in quotation marks will generally be processed as word separators, causing the original literal to be divided into two or more literals. In addition, the special character may be processed as a literal

by itself. The geographic operators expect literals in a special format as described in the following paragraphs.

i. Coordinate/Radius (required for CIR) -- is a literal of a coordinate in 11- or 15-character format, i.e.,

DDMMXDDDMMY/XXXXX
DDMMSSXDDDMMSSY/XXXXX

where

DD	=	latitude in degrees
MM	=	minutes of a degree
SS	=	seconds of a minute
X	=	N for North and S for South
DDD	=	longitude in degrees
Y	=	E for East and W for West
/	=	subfield separation symbol
XXXXX	=	radius in nautical miles not to exceed 10800.

The coordinate will be converted automatically to internal format by a standard system subroutine (UTCORDI).

j. Coordinate/Coordinate... (required for OVP) -- is a literal composed of one or more coordinates separated by slashes and describes a point, line, or polygon. Vertices must be listed in counterclockwise order. No polygon may overlap the Greenwich Meridian (0 degrees longitude). The circle search and overlap relationship operators may be used only with coordinate fields.

k. The BT (BETWEEN) relational operator requires the literal to be two literal values separated by a slash. Any number of literal pairs may be specified.

The use of a dollar sign ($) in a literal not enclosed in single quotes indicates a universal match (see section 3.7, Universal Match Character) and causes that character

position to be ignored when compared to its corresponding character in the data base field. Dollar signs may not be used with the relational operator CONTAINS, geographic operators or literals which are to be converted by a subroutine. The use of a dollar sign as a universal match character within a clause will prohibit Secondary Indexing for that clause.

Multiple literals may be used with the EQ, NE, CONTAINS and BT relational operators. When using the NE relational operator, there is an implied AND-type logical connector between the literals. With the EQ and BT relational operators, there is an implied OR-type logic connector between the literals.

The following is a list of some of the conditions that will be checked for during the translation of the retrieval type IF operator:

a. An unlimited number and level of parentheses will be allowed.

b. Coordinate fields and literals with operators other than OVP and CIR may compare latitude-to-latitude or longitude-to-longitude only; point-to-point comparison is forbidden.

c. Only one object is allowed in field-to-field comparison except for the BT operator.

d. In field-to-field comparison, no more than two sets may be referenced; if two are referenced, one must be the fixed set.

e. No data value is allowed with the ABSENT operator.

Example 1

```
IF  UIC 1/1 #RCMDS# EQ USAF AND MEQPT EQ F-111 AND MEPSD
EQ &MEOFC.
```

Comments: Illustrates the use of partial field notation, subroutine conversion, and an indirect address literal to

retrieve Air Force records having F-111s as their major equipment.

Example 2

```
IF SERV NE ARMY NAVY.
```

Comments: Illustrates the use of multiple literals and automatic table conversion. This condition is satisfied if some value other than ARMY or NAVY is contained in the service field. The example assumes that the input table conversion RCMDS has been specified for the field SERV in the File Format Table (FFT).

Example 3

```
IF POINT CIR 4851N00220E/225.
```

Comments: The CIR (circle search) relational operator is used to test if a point falls within a radius of 225 miles of Paris, France.

Example 4

```
IF UNTYP EQ 24VC$, 24VS$.
```

is valid as is the following example:

```
IF UNTYP EQ $4V$C, $4V$S.
```

Comments: When using multiple data values and the dollar sign($) feature, leading, trailing and imbedded dollar signs ($) are permitted in alpha literals. Only leading and trailing dollar signs are permitted in decimal literals.

Example 5

```
IF COMMENT CONTAINS VIET.
```

In the above example, a data record will qualify if the pattern VIET occurs anywhere in the variable length field COMMENT.

Example 6

```
IF MEQPT EQ F-111.
```

In the above example a data record will qualify if a periodic subset has the value F-111 in the periodic field MEQPT. Additionally, the qualifying subset will be "flagged" and its fields along with the FIXED fields will be available for processing by the LIST, COMPUTE, COUNT, EXTRACT, SUM, and DISPLAY operators. The nonqualifying subsets are not available for output processing unless the 'ALL' modifier precedes the field name (see section 3.6.1).

Example 7

```
IF MEQPT ABSENT
```

In the above example a data record will qualify if the periodic set having a field MEQPT has no subsets. The 'NOT' modifier may reverse the logic to qualify records with subsets containing a field MEQPT, e.g.

```
IF MEQPT NOT ABSENT
```

(Note: in the last example the 'NOT ABSENT' test does not flag subsets when a record qualifies. The 'ALL' modifier must be used when fields from the periodic set are referred to in output processing statements. (see section 3.6.1).

Example 8

```
IF ATACH EQ SAC AND

((GEPOL EQ NA AND CNTRY NE CN AND MEDEP EQ ' ')

OR (MEPOL EQ NA AND METRY NE CN AND MEDEP EQ '  '))

AND MECL EQ A$ M$.
```

Comments: This illustrates the use of nested parentheses. The condition requests logical blocks for units whose status is reported by the Strategic Air Command. The unit must be located in the North American Continent, but not in Canada,

and have no major equipment deployed; or the geopolitical area in which the unit may be deployed must be in the North American Continent, but not in Canada, and it must not have major equipment deployed currently. In either case, the major equipment class must begin with the letter A or M.

Example 9

```
IF MEPNT OVP 4900N00650E/4940N00615E/4945N00220E.
```

Comments: This illustrates the use of the polygon search capability to locate units whose major equipment is deployed within a selected area. The coordinates must be defined in a counterclockwise order, and no more than eight may be specified.

3.5.9.1.1 FUNCTION Operator

In addition to the standard query language discussed in the previous section, the user may write his own subroutine to assist in record qualification and data presentation. (See Volume IV, Retrieval and Sort Processor, section 2.4.2.3.4 FUNCTION Operator and section 2.8 Writing a FUNCTION Operator Subroutine for further information).

The FUNCTION operator is a special operator written as a conditional clause within the statement formed by the retrieval-type IF operator. The FUNCTION operator permits the execution of a user-written subroutine as part of the qualification of a data record. It will normally be preceded and followed by a logical connector and another conditional clause. Use of this operator is designated by the keyword FUNCTION, followed by the name of the function; e.g., the user's subroutine name. All terms following the function name and preceding the next logical connector or the end of the statement will be considered as part of the parameter list to the function subroutine. Secondary Index processing can also be performed for the FUNCTION operator under the conditions described in section 3.5.9.3, Retrieval Operators and Secondary Indexing. The format of the clause formed by the FUNCTION operator is as follows:

```
FUNCTION    subroutine name    parameter(s)
```

The parameters may be literals, indirect address literals (field names prefixed with an ampersand) or function work areas (WORK1 through WORK5, see section 3.5.9.1.2).

There are several rules or restrictions which the user must follow to use the FUNCTION operator. Basically, the FUNCTION operator and any correlation between subroutine requirements and designated input are the user's responsibility. If the following rules are heeded, there should be little problem using the FUNCTION operator:

a. A double comma cannot be used to indicate omission of a parameter.

b. Parameters must be designated in the order prescribed by the subroutine.

c. Literal values in the parameter list are subroutine related and not data field related; thus, there will not be any automatic conversion by subroutines of literals on input.

d. A literal containing embedded blanks or commas must be enclosed in quotes.

e. A literal containing any nonnumeric character will be passed to the function subroutine as EBCDIC characters.

f. A literal containing all numeric characters will be converted to a binary value prior to being passed to the function subroutine. Enclosure of an all-numeric value in quotes will not prevent that value from being converted to binary; i.e., a literal must contain a nonnumeric character to be passed to the function subroutine as an alpha literal.

g. Data file field names must be prefixed with an ampersand.

h. The parameter list to the FUNCTION operator subroutine cannot contain the words AND or OR.

1. Partial-field notation or user subroutine conversion will not be allowed with items in the parameter list.

Example

```
IF SERV EQ NAVY AND FUNCTION SUPRFNC &PERS.
```

Comments: The user subroutine (SUPRFNC) could be used to provide further qualification of the NAVY records based on the PERS file field.

3.5.9.1.2 Function Work Areas

There are five 4-byte work areas available to function subroutines. The contents of these work areas are never changed by QUIP. They are available to the subroutine to generate data for use by the other operators (LIST, DISPLAY, COMPUTE, etc.). Work areas may be associated with any set of a data record and may have different output lengths and data modes (alpha, binary, coord, and decimal). The associated set number, output length, and data mode must be specified within parentheses following the work area name each time the work area is used with an output operator as follows:

WORKn (s,l,m)

n - work area number (1,2,3,4 or 5)

s - set number (0 - 255), default to 0

l - output length (1 - 10), default to 10

m - data mode (A,B,C,D), default to B

If the default values are satisfactory, the entire parenthetical expression may be omitted. The terms within the parentheses are positional; i.e., if only a different length is desired, a comma must precede it.

Example 1

```
WORK2 (,6)
```

Comments: WORK2 would be associated with the fixed set, binary data mode with an output length of 6.

Work areas may be used with all QUIP operators wherever file field names are valid.

Example 2

```
IF SERV EQ NAVY AND FUNCTION SUPRFNC &PERS WORK1.

LIST UIC PERS WORK1 (,8)
```

Comments: In addition to contributing to data record qualification, the user subroutine, SUPRFNC, passes binary data of output length 8 for display by the LIST operator.

3.5.9.2 FIND Operator

The FIND operator allows the user to specify the number of records to be retrieved before end-of-file is reported. It may be followed by a numeric literal indicating the number of records to be retrieved. If the numeric literal is not present, 1 will be assumed as the default value.

The FIND operator may also be used to specify the number of qualifying records to be retrieved. Used in this manner it has the same format as the retrieval type IF except that a numeric literal, indicating the number of qualifying records, may precede the first conditional clause. Secondary Index processing can be applied to this operator as described in section 3.5.9.3, Retrieval Operators and Secondary Indexing.

The following restrictions are imposed on the FIND operator:

a. The number of records specified by the FIND operator is constrained by any bounds established through the use of the LIMIT operator (see section 3.5.10, LIMIT Operator).

b. Only one FIND or retrieval type IF operator is permitted within a query.

c. The FIND operator is not allowed within the range of an EXTRACT operator.

The FIND operator must be terminated with a period and has the following format:

```
FIND [n] [ANY] {fieldname}                                    {geo. op }
               {groupname} [m/n][#subtab#][NOT] {rel. op } literal(s)
                                                conditional clause

   ... {AND} conditional clause .
       {OR }
```

a. n is a numeric literal which specifies the number of records to be retrieved.

b. See section 3.5.9.1, Retrieval Type IF Operator, for a discussion of the retrieval conditional clause.

Example 1

```
FIND 50.
```

Comments: The first 50 records passing the LIMIT test are retrieved.

Example 2

```
FIND 10 CNAM EQ FRANCE AND LOC EQ PARIS.
```

Comments: Retrieves only the first 10 records which pass the LIMIT test and are located in Paris, France.

3.5.9.3 Retrieval Operators and Secondary Indexing

Secondary Indexing is an automatic feature of QUIP intended to provide a faster and more efficient retrieval. An indexed retrieval operates in the candidate-access mode processing only the records Index Processing has listed as possible candidates. A nonindexed retrieval operates in the file-access mode processing the complete file except as restricted by subset qualification or a LIMIT statement.

Indexing will be attempted if the user has not specified INDEX=NO in the PARM entry of the EXEC XQUIPSD card, if the input source statements include retrieval logic, and if the input file is indexed. (Refer to Volume II, File Structuring, for details of indexing a file.) If any clause within the retrieval type IF statement or the FIND statement uses partial field notation, field-to-field comparisons, the CONTAINS or ABSENT relational operator, or the universal match character ($), the clause will not qualify for Index Processing.

If Secondary Indexing is attempted, QUIP calls the Index Processing function to analyze the user's retrieval logic and determine if index usage is feasible.

3.5.9.3.1 Retrieval Type IF Operator and Secondary Indexing

This section explains how, once invoked, Index Processing determines if the indexing information can be utilized to direct QUIP in accessing the data records.

Index Processing bases its decision on the presence of an indexed field in at least one "must-satisfy" clause of the retrieval. A must-satisfy clause is defined as a clause whose condition must be true in the data record in order for that record to qualify for retrieval. Whether a clause is in the must-satisfy state or not depends on how the retrieval is written. Therefore, the decision to utilize index information or not is always based on the dynamic structure of a retrieval at run time.

The basic criterion in deciding whether a clause is a must-satisfy or not is if the clause (or string or group of clauses) is connected to a clause of equivalent logical level by AND. Connection by AND ensures that the clause condition must be satisfied in order to retrieve the record. Connection by OR gives the processor a choice of terms by which to qualify the record. Example:

```
IF CNTRY EQUAL US.
IF CNTRY EQUAL US AND MEQPT EQUAL TANK.
```

Comment: In each of these statements, the clause CNTRY EQUAL US is a must-satisfy clause, because this condition must be true in order to retrieve the record.

Contrast the above examples with this:

IF CNTRY EQUAL US OR SERV EQUAL ARMY.

Comment: In this case, CNTRY EQUAL US is not in the must-satisfy state. If this condition is not true, QUIP can still qualify the record if SERV is equal to ARMY, irrespective of the field CNTRY contents.

Refer to Volume II, File Structuring, Section 3.3, Sample File Structure Job, for a description of the TEST360 file. Note that SERV and CNTRY are fixed field indexes, and MEDDT and METRY are Periodic Set 1 indexes. PLRT, a periodic field in Set 3, and SPCODE, a periodic field in Set 5, complete the list of indexes.

In the following examples, the operator is EQUAL; however, any other valid operator could be used as well. An index, fixed or periodic, may be used. Qualification is at the data record level, not at set level.

Example 1

IF CNTRY EQ GY.

Comment: This example shows a retrieval with one clause. Only the records whose field CNTRY equals GY will be presented as candidates. In this example, of course, all candidates are also qualifiers.

Example 2

IF PERS GT 10000 AND CNTRY EQ UK.

Comment: Each clause is a must-satisfy clause. That is, PERS must have a value greater than 10000 and in addition the country must equal UK. As CNTRY is indexed the search for qualifying records can be restricted to those records whose field CNTRY contains the value UK.

Example 3

```
IF CNTRY EQ GY OR CNTRY EQ FR.
```

Comment: The search would be restricted to records containing GY or FR in the field CNTRY.

Example 4

```
IF CNTRY EQ GY OR PERS GT 10000 AND CNTRY EQ UK.
```

Comment: Records containing GY as the CNTRY, or records containing UK as the CNTRY and more than 10000 in PERS would qualify. Index Processing would restrict the search to records carrying GY or UK in CNTRY.

Example 5

```
IF (CNTRY EQ GY OR CNTRY EQ UK) AND PERS GT  10000.
```

Comment: Although the logic of this example would restrict qualifying records to those with GY or UK in CNTRY and PERS greater than 10,000, Index Processing would present the same records as in example 4.

Example 6

```
IF (SERV EQ ARMY AND UNRDY EQ A) OR (REQPT EQ A AND
METRY EQ US).
```

Comment: In each parenthetical group, an indexed field is referenced in a must-satisfy clause. Since the qualifying records could contain either combination of values as expressed in each group, retrieval must search all records in which SERV equals ARMY or METRY equals US. Both could of course be present in one record, but Index Processing would only be concerned with the either/or case. All other records would be excluded from the detailed search.

Example 7

```
IF  UNTYY NE ARTL AND (SERV EQ ARMY OR METRY EQ US)
AND UNTYZ EQ X.
```

Comment: The logic of this example would restrict the qualifying records to those whose UNTYY field was not ARTL and whose UNTYZ field was X. In addition, the SERV field must equal ARMY or the METRY field must equal US. As the group is a must-qualify group, one clause or the other must be true in order for the data record to qualify. Therefore, Index Processing would determine that only records whose field SERV contained ARMY or whose field METRY contained US could possibly qualify. In this manner, all other records would be eliminated from the detailed search.

3.5.9.3.2 FUNCTION Operator and Secondary Indexing

The FUNCTION Operator parameter string will be examined to determine if a data file field, or fields (a data field name must be prefixed by an ampersand), is present. If so, and any field so referenced is indexed, Index Processing will be invoked. If no field name is given in the FUNCTION Operator parameter string, or no field is an index field, index usage will not be possible for that clause. Only one indexed field per function clause (the first one encountered) can trigger index usage. Any other indexed fields are treated as nonindexed fields in the analysis.

The indexed field used in a FUNCTION operator clause must have an analyzer routine specified for it. An analyzer routine is designated for a field in an Index Specification run (reference Volume II, File Structuring). The purpose of the analyzer routine is to analyze the FUNCTION operator parameters for Index Processing. The analyzer routine will return a value (or values) which Index Processing will use as arguments when searching the field's index information in the Index Data Set. This search is conducted in an "equal" relationship only. If no analyzer routine is specified for any index field, Index Processing will be negated for the entire FUNCTION operator clause.

3.5.9.3.3 FIND Operator and Secondary Indexing

The Index Processing function treats a FIND operator as a retrieval type IF and the numeric literal preceding the first clause is ignored.

3.5.10 LIMIT Operator

The LIMIT operator allows the user to limit the records read from the file to specified values of the key (or high-order portion thereof). It is not allowed within the range of an EXTRACT operator. This operator should be terminated with a period and has the following format:

```
        {fieldname}                          relational
LIMIT   {groupname} [m/n][#subtab#] operator    literal.
```

a. The field or group, if specified, must include the high-order character(s) of the major control field. When no field or group is specified, QUIP will assume a default of the complete record key.

b. m/n indicates partial field notation and specifies which portions of the record key will be used for the comparison. The partial field must start at the first character; i.e., 1/n.

c. The field or group may be modified by a subroutine expression. Double pound signs (##) suppress automatic table conversion, and the name of a subroutine enclosed in pound signs forces table conversion.

d. The relational operators shown below are allowed in the statement formed by the LIMIT operator and condition the selection of records as follows:

EQ - search when equal to

LT - search when less than

LE - search when less than or equal to

GT - search when greater than

GE - search when equal to or greater than

BT - search when equal to or between.

The logical connector NOT may precede all relational operators except EQ and BT in a LIMIT statement.

e. The literals will be converted automatically to their coded form if an input subroutine or table was specified for the field in the File Format Table (FFT). When no field or group is specified, the literal will not be converted to a coded form. If this is desired the user must specify the input conversion table or subroutine.

Example 1

```
LIMIT UIC 1/1 BT M/N.
```

Comments: The search of the file is limited to MARINE and NAVY records. Note that the BT relational operator must be followed by two literals separated by a slash.

Example 2

```
LIMIT SERV EQ ARMY.
```

Comments: The literal ARMY will be converted automatically by the RCMDS input table and only those records containing a 'W' in the high-order position of the record ID field will be processed.

Example 3

```
LIMIT BT M00001/M00500.
```

Comments: The search of the file will be limited to those records with record keys containing the values M00001 throught M00500 in the high-order 6 positions.

3.5.11 KEYWORD Operator

The KEYWORD operator allows the user to query fields or variable sets that he has previously designated as keyword indexed fields. Qualification is on the record level. The KEYWORD statement consists of conditional clauses, logical connectors and parentheses.

The format is:

```
                    AND                 AND
KEYWORD  clause      OR  clause...      OR  clause.
```

where

KEYWORD (required) - is the statement identifier and must be the first work in the statement.

AND, OR (optional) - defines the relation of two clauses when the KEYWORD statement contains more than a single clause.

Clause (required) - is a true assertion and must use INCLUDES as the relational operator. The method of coding clauses is defined in the following section.

. (required) - statement terminator.

3.5.11.1 Condition Clauses and Keyword Indexing

The condition clause in a KEYWORD statement adheres to the same logic as the standard condition clause of an IF statement only with fewer variables. The complete format of one clause is:

```
                                 data values
     fieldname    INCLUDES       literals
```

where,

fieldname (required) - identifies the name of a fixed-length alpha field, variable field or variable set that has been defined as a keyword field and is to be tested against the data value.

INCLUDES (required) - is the relational operator stating the data value must be contained within the specified field.

Data value, literal(required) - adheres to the same definitions established for them in the IF statement.

In order for the retrieval to qualify, this value must have been selected as a keyword for this field.

Partial-field notation and data value conversion subroutines are not permitted. Data for keyword fields is carried in the file in the same format as it was entered.

There may be multiple data values or literals specified and there may be any number of clauses.

All specified fields on the KEYWORD statement must have been defined as keyword indexes.

If a "must-satisfy" clause produces no qualifying records all retrieval processing is terminated with an advisory message.

The qualifying records from the KEYWORD statement are merged with the secondary indexed qualifying records from the IF statement to produce a final candidate list for retrieval. If secondary indexing was not applied (or proved unfeasible) the keyword list becomes the final one. But if no candidates were produced from secondary index processing, the retrieval is terminated.

3.5.11.2 Examples of Keyword Indexing

Prior to issuing a query containing a KEYWORD statement, the user must have defined the variable fields, variable sets or fixed-length alpha fields he desires to query as keyword fields. He may do this through File Structuring (Volume II, Index Definition for Keyword Fields section) or through the Index Specification Utility (Volume VII, Section 12). This process creates an index data set for the data file containing the designated fields, the keywords within the fields and the corresponding record IDs.

This method offers a fast and efficient retrieval in that only the records known to contain the desired keyword will be accessed.

The KEYWORD statement consists of one or more clauses. Each clause may contain one or more data values.

Example 1

```
KEYWORD   REMARK   INCLUDES   NMCSSC
```

In this example, REMARK is a variable field in the data file that has also been defined as a keyword index. Only those records that have a REMARK field containing NMCSSC will be presented as candidates for QUIP retrieval.

Example 2

```
KEYWORD REMARK INCLUDES NMCSSC, AOC OR REFER INCLUDES
'NOVEMBER,1972'.
```

In this example, REMARK is a variable field, REFER is a variable set, and both have been defined as keyword indexes. All records that have a REMARK field containing either NMCSSC or AOC or a REFER containing NOVEMBER,1972 will be presented as candidates. This literal must be bounded by quotes in order to have the special character of a comma included as part of the value search.

Example 3

```
KEYWORD (REMARK INCLUDES VIETNAM AND TYPE INCLUDES F-111)
OR (COMMENT INCLUDES AIRCRAFT AND SERV INCLUDES N).
IF DATE EQ JUL72.
```

In this example, REMARK and COMMENT are variable fields, TYPE and SERV are fixed-length alpha fields, and all have been defined as keyword indexes. DATE is a fixed-length alpha field defined as a secondary index. The following records would become candidates for retrieval: those having a REMARK containing VIETNAM and a TYPE containing F-111 and a DATE equal to JUL72; those having COMMENT containing AIRCRAFT and SERV containing N and DATE equal JUL72.

Example 4

```
KEYWORD REMARK INCLUDES TECHNICIAN.
IF RANK EQ 'E-7'.
```

In this example, RANK is not an indexed field. The candidate list would consist of those records containing a

REMARK field which includes the value TECHNICIAN. These records will be accessed by the QUIP retriever to examine the RANK field. The qualifying retrieval records will be only those that have both REMARK containing TECHNICIAN and RANK equal E-7.

Example 5

```
KEYWORD COMMENT INCLUDES NEW AND COMMENT INCLUDES
CAPABILITY.
IF DATE EQ SEP72.
```

In this example, both fields have been specified as being indexed. Each has qualifying candidates, but there are not candidate records common to both conditions, i.e., no record that has a COMMENT field including NEW and CAPABILITY has a DATE field equal to SEP72 or vice versa. In this case, retrieval will be terminated with an advisory message. This would also happen if either of those fields had no candidate records.

Example 6

```
LIMIT DATE BT JUL71/JUL72.
KEYWORD COMMENT INCLUDES NEW.
IF TYPE EQ K.
```

TYPE has not been defined as an indexed field. LIMIT processing will eliminate all records that have a DATE field not between JUL71 and JUL72. Index processing will present for retrieval a list of record-IDs whose records have a DATE between JUL71 and JUL72 and a COMMENT field containing NEW. Final qualifications will be those records that also have a TYPE field equal to K.

3.5.12 SORT Operator

The SORT operator permits the user to specify the fields which are to be used to sequence the retrieved records. It also allows the user to assign a new sequence to the data file records for output purposes. The optional flag #D# permits the user to specify the fields which are to be ordered in descending sequence. When the SORT operator is used, a sort key, which may hold up to 210 characters of

user-supplied information, is generated. The user may reference data in the sort key during output by use of the keyword SORTKEY. He may also display the contents of the sort key, or portions thereof, by using this keyword in a DISPLAY operator.

The SORT operator, which should be terminated with a period, has the following format:

$$\text{SORT} \left\{\begin{matrix}\text{fieldname} \\ \text{groupname}\end{matrix}\right\}[(m/n)] \left[\begin{matrix}\#D\# \\ \text{or} \\ \#\text{subtab}\#\end{matrix}\right] \ldots \left\{\begin{matrix}\text{fieldname} \\ \text{groupname}\end{matrix}\right\}[(m/n)] \left[\begin{matrix}\#D\# \\ \text{or} \\ \#\text{subtab}\#\end{matrix}\right]$$

a. The fields/groups may be fixed or periodic. However, all periodic fields/groups must be from the same set and that set must have been queried in a previous retrieval type IF or FIND statement. The presence of a periodic field/group in a SORT statement will cause QUIP to build a sort key for each qualifying subset in which the periodic field/group appears. Therefore, it is possible for multiple sort keys to be built for a single NIPS record. If the periodic set was qualified by the ABSENT operator, attempts to sort records on periodic fields from the referenced set will cause these answer entries to be eliminated. The records are excluded because there are no flagged subsets on which to sort the records. If output table conversion was specified for a field/group in the FFT, that field/group will be converted automatically before it is placed in the sort key.

b. Partial field notation may follow a field or group name and designates that portion of the field or group that is to be inserted in the sort key. If used with a subroutine expression, it must precede the conversion specification.

c. A subroutine expression can be used to modify a field or group name. A double pound sign (##) will suppress automatic conversion, and a subroutine name enclosed in pound signs will force table conversion.

d. The use of the reserved notation #D# will cause the associated field/group to be inserted in the sort key in inverted form, thus providing inverse ordering of field/group values. This notation is valid where subroutine conversion is allowed and partial field notation may also be specified. Display of the sort key contents for these positions will cause nonprintable characters to be displayed.

The following list of rules will be used during the translation of the SORT operator:

a. Only one periodic set may be specified, and it must have been referenced in the retrieval IF operator.

b. Total user key - 210 bytes.

c. Coordinate fields = latitude, longitude or point (no areas allowed).

d. Binary/coordinate fields will be inserted using external format and length.

e. Binary/alpha/decimal fields will be inserted in inverted form when #D# notation is used.

f. The SORT operator or reference to the sort key is not allowed within the range of an EXTRACT operator.

g. Only one SORT statement is allowed during an execution of QUIP. It is not allowed if the input to QUIP is a RASP Answer Set.

h. Only fixed length fields may be referenced by the SORT operator.

Example 1

```
SORT LOC 1/16 SERV ##.
```

Comments: This illustrates the use of partial field notation and output table suppression as applied to different fields. The records available for output will be sequenced by location and service code within location.

Example 2

```
SORT UIC 1/1 #OCMDS#.
```

Comments: When partial field notation and subroutine conversion are applied to the same field, the partial field expression must precede the subroutine expression. The first character of the UIC control group (the field SERV) will be passed through the OCMDS output table before it is placed in the sort key.

3.5.12.1 SORT Operator with Sort Space Allocation Modifier

In addition to allowing the user to specify the sequence of the retrieved records, the SORT operator also provides the user with the capability to adjust the primary amount of temporary direct access storage space dynamically allocated for retrieved records and sortkeys during online retrieval. This is accomplished by permitting users to include sort space allocation modifiers. There are two such modifiers, one for the retrieved records and one for the sortkeys. A modifier must be enclosed in parentheses, and the format is:

```
[RECS],value   or   value,[RECS]
[KEYS]                 [KEYS]
```

The above identifies that either retrieved records or sortkey space is to be increased during online retrieval.

Value - A plus or minus 1- to 3-digit number reflecting space units that will be added to the internally programmed primary space allocation figure. (The unit of

space is determined by JCL DD card SYSUT2, and can be cylinders or tracks.)

Note that the sequence of the sort modifiers is unimportant and that plus values can be signed or unsigned.

If the value specified by the user to reduce a programmed primary space quantity is larger than the programmed value, QUIP will set the value to 1.

Either, both, or none, of the space modifiers can be included anywhere between the SORT operator and the terminator(.). The modifiers may even be intermixed with the sort fields without affecting the sortkey sequence. In the examples that follow, FLD1, FLD2 and FLD3 represent file fields to be inserted in the sortkey.

Example 1

```
SORT (20,KEYS),FLD1,FLD2,(RECS,-10).
```

In the above example, 20 extra space units will be added to the internally programmed primary space allocation figure for the sortkeys, and 10 space units will be subtracted from the allocated space for the retrieved records.

Example 2

```
SORT (RECS,5),(KEYS,10) FLD1.
```

In the above example, 5 extra space units will be added to the internally programmed primary space allocation figure for the retrieved records and 10 space units will be added to the allocated space for the sortkeys.

Example 3

```
SORT FLD1,(RECS,8),FLD2,(KEYS,2).
```

In the above example, eight extra space units will be added to the internally programmed primary space figure for the retrieved records and two space units will be added to the allocated space for the sortkeys.

Example 4

```
SORT FLD3,FLD1.
```

In the above example, no additional units of space will be added to the internally programmed primary space allocation figure for the sortkeys or retrieved records.

#Note - The sort space allocation modifier will be accepted and checked for errors in either the batch mode or online mode, but it is effective only in the online mode. Space is not allocated dynamically in the batch mode; it is determined by the JCL DD card specification.

3.5.13 Library Action Operators

The operators in this category permit the user to execute from the library a query previously stored by QUIP and/or a RIT previously stored by the Output Processor component. Only one library action operator is allowed in the user's input stream. If the user's input stream includes a LOAD QUERY operator, the stored query may contain a LOAD RIT operator.

QUIP queries may be stored on the file library or a source library containing 80 character card images. Only the QUIP component, the UTSOURC utility or online TPEDIT capability may be used to store or replace queries on the file library. A complete discussion of source library storage is contained in the Utility Support User Manual, Volume VII, Section 10, Source Language Storage.

3.5.13.1 LOAD QUERY Operator

The LOAD QUERY operator may be used to load a stored query from the user's library and execute it. This operator must be the last operator in the input stream and must be terminated with a period.

When executing a LOAD QUERY in the batch mode environment source statements may begin in any column and continue through column 71 (see para 3.3 for source language formatting). When executing a LOAD QUERY in the online

environment, source statements may be expressed in any column between 1 and 80.

QUIP will effectively concatenate the stored query with any operators already entered in the input stream. This capability permits specification of the FILE (QUERY) operator at the terminal. The operators in the stored query in combination with those in the input stream may contain only one retrieval type IF or FIND operator.

The format of the LOAD QUERY operator is described below:

LOAD QUERY 'query-name' ['libname'] [(keyword=replacement variable)].

a. The query-name is a 1- to 7-character name assigned to the query when it is stored.

b. The libname is the name of the user library containing the stored query. This name must be enclosed in apostrophes. The library name must be specified if the query is not in the file library when QUIP is executed from the terminal. When executing QUIP in the background environment, however, the library must be specified in the JCL on the SLIB DD statement.

c. The LOAD QUERY operator permits the definition of replacement variables when executing a Skeleton Query (see Section 3.8, Skeleton Query). The replacements are enclosed in parentheses and are placed as the last set of operands in the statement. Each word in the stored query which is preceded by the underscore character (0-5-8 punch) is eligible for replacement (i.e., a replaceable variable). The replaceable variable is used as a keyword. This is followed by an equal character (=) and the replacement variable. There are no spaces between the keyword, the equal character, and the replacement variable. When an operand is being replaced by multiple operands, the operands must be separated by commas or blanks and must be enclosed in parentheses within the parentheses of the entire replacement string. All replaceable

variables that have not been assigned replacements will default to their original value.

Example 1

```
LOAD QUERY 'QUERY1'.
```

Comments: The query named QUERY1 will be loaded from the library defined in the JCL parameters for the run.

Example 2

```
LOAD 'QUERY2' 'TEST360L' (ARMY=(NAVY MARINE),PARIS=TOULON).
```

Comments: The query named QUERY2 will be loaded from the library named TEST360L. As the query is being scanned, every underscored occurrence of the word ARMY will be replaced by the values NAVY and MARINE; and every underscored occurrence of the word PARIS will be replaced by the value TOULON.

3.5.13.2 LOAD RIT Operator

The LOAD RIT operator is used to execute a RIT from the user's library to format the output display. It is not allowed within the range of an EXTRACT operator. When this operator is used, only the following QUIP operators will be executed:

```
CLASS
FILE
FIND
IF (Retrieval type)
LIMIT
LOAD QUERY
NLINES
SORT
```

All other operators are ignored. The LOAD RIT operator can be used with the LOAD QUERY operator only when the LOAD RIT operator is part of the stored query which is loaded.

The format of the LOAD RIT operator is:

```
LOAD RIT 'ritname' ['libname'] [PARAM='parm list']
         [EXTRACT={filename}].
                  {NO      }
```

a. The ritname is a 1- to 7-character name assigned to the RIT at OP structure time. The name must be enclosed in apostrophes.

b. The libname is the name of the user library containing the stored RIT. This name must be enclosed in apostrophes. The library name must be specified if the RIT is not in the file library when QUIP is executed from the terminal. When executing QUIP in the background environment, however, the library must be specified in the JCL on the SLIB DD statement.

c. The LOAD RIT operator will permit a parameter list to be introduced to OP at execution time. This parameter list is identified by the keyword PARAM, followed by an equal character (=), and must be enclosed in apostrophes.

d. The EXTRACT operand may be used for an RIT structured for IFO to limit the secondary files used by the query. If all secondary files referenced in the RIT are desired, the operand is not required. Otherwise, each secondary file that is required for processing must be entered following the equal sign. If more than one file is entered, the group of names must be enclosed in parentheses and separated by commas or blanks. If no secondary file processing is required, the keyword NO is entered to indicate that only the primary file is to be processed. RIT specifications for secondary files not entered with the EXTRACT operand are ignored by the query.

e. The LOAD RIT operator must be terminated with a period.

Example 1

```
LOAD RIT 'RITONE'.
```

Comments: A RIT named RITONE will be used by QUIP to format the output.

Example 2

```
LOAD RIT 'OPRIT' 'TEST360L' PARAM='MONTHLY'.
```

Comments: This example will cause QUIP to use the RIT named OPRIT, which is stored on a library named TEST360L, to format the output display. The word 'MONTHLY' is passed to the RIT to be used at RIT execution time.

Example 3

```
LOAD RIT 'IFORIT' EXTRACT=(FILEA,FILEB).
```

Comments: The RIT named IFORIT which was structured for IFO is to be used to format the output. Any EXTRACT statements in the RIT referencing any secondary files other than FILEA or FILEB will be ignored.

3.5.14 Terminal Environment Operators

The seven operators SIGNON, SUBFILE, SIGNOFF, TRACE, FORMAT, VIEW and OMQ are applicable only to the online environment.

3.5.14.1 SIGNON Operator

The SIGNON operator is used in the online environment to identify to QUIP the requirement to pass the FILE and CLASS operator information to succeeding queries. It has no effect if it is used in the background environment.

The SIGNON operator without operands may appear as the only operator, or it may be included as an operator within a query. In any event, it must appear in or before a query that identifies a file which will be interrogated by

following queries which are not required to include FILE and CLASS operators.

The SIGNON operator places QUIP in signon mode, which is in effect until canceled by a SIGNOFF operator in some later query. For this reason SIGNON need not be repeated unless a SIGNOFF operator has removed QUIP from signon mode or a restart of the TP system is required.

When the signon mode is in effect, any query (other than the query which first identifies the file) which does not include a FILE operator is processed using the same input source as the previous query, providing the previous query did not create a subfile. Thus, in signon mode the FILE operator is required only for the first interrogation of a file in signon mode or when a change of input source is required.

The SIGNON operator has an optional name operand which may be used in connection with the subfile function in the event a restart of the TP system is required. It allows an existing subfile library to be made available when the TP system is restarted after an abnormal termination of the TP control program or an operating system failure, providing the failure was of such a nature that retention of the subfile library was possible. It also obtains the FILE and CLASS operator information for the master file. When the name operand is included, the signon operator must be the only operator in the query. That query may then be followed either by a query which accesses the master file as the input source, in which case FILE and CLASS operators are not required, or by a query which references the desired subfile as the input source, thereby restarting the subfile function at that time.

While QUIP is in the signon mode, any query which has a syntax error or any other type of error which prevents the successful completion of the query causes QUIP to be restored to its status prior to the start of the query in error except that QUIP remains in signon mode even if a query which contains the SIGNON operator has errors. This allows continuation without any adverse effects as a result of the error.

The format of the SIGNON operator is:

SIGNON [name]

a. name is the optional operand used to identify a previously created subfile library which is to be made available for continued subfile processing when restart of the TP system is required. The name must be the same as the one internally assigned to the library by QUIP at the time it was allocated in an interrogation session which was interrupted due to abnormal termination of the control program, an operating system failure, or a machine malfunction. The status of the subfile library when it is restored to availability by the SIGNON operator is that which existed after the last successful query before interruption occurred. When the operand is included, the SIGNON operator is the only operator allowed in the query.

Example 1:

SIGNON FILE filename. CLASS classification LIST field

Comments: The SIGNON operator in this online query identifies the file which may be interrogated in subsequent queries which need not contain the FILE or CLASS operators. At the completion of this query QUIP would be in signon mode even if syntax or execution errors occurred during processing.

Example 2:

NMCSSC1.131302 or
SIGNON SYS75302.T131302.RV000.NIPSMTP.SUBFILE

NOTE: The format of the QUIP assigned subfile library name is determined by the option specified when the NIPS TP Supervisor is generated. Section 7.5 of the NIPS Installation Manual details the available options.

Comments: This query might be the first query entered when a restart of the TP system occurred after a previous

terminal session which created subfiles was interrupted. It identifies the subfile library with the data set name NMCSSC1.131302. QUIP then attempts to locate the library and make it available for continued subfile processing. NMCSSC1.131302 is the name assigned to the library by QUIP when it was allocated. This query must be followed by a query against the master file to allow location and setup of the master file prior to any subfile processing.

3.5.14.2 SUBFILE Operator

The SUBFILE operator provides the capability in the online environment to create and subsequently interrogate a subfile of a NIPS file. The subfile created consists of selected entries retrieved from the file (called the master file in subfile context). The entries in the subfile are selected based on the conditional expressions in the retrieval and/or limit logic presented by means of retrieval - IF, FIND, LIMIT, and/or KEYWORD operators. At least one of these operators must be present in a query which is to create a subfile. Subsequent queries are automatically directed against the subfile until one of the following occurs:

a. the user creates a lower level subfile causing subsequent queries to be directed against the new subfile;

b. the user changes to another NIPS file, the master file, or a previously created subfile by means of the FILE operator causing subsequent queries to be directed against the new input source;

c. the user signs off from QUIP with the SIGNOFF operator causing any subsequent query to require a FILE operator (see section 3.5.14.3, SIGNOFF Operator).

There are no restrictions on the number of subfile levels which may be created. Any previous subfile or the master file may be specified on a FILE operator in subsequent queries so that interrogations may be performed from that level. However, once the SUBFILE operator has been specified, any query which is not directed against a

subfile or the master file causes the subfile library to be logically deleted, thereby making any subfiles which may have been created up to that time unavailable for further processing. Thus, subfiles may exist for only one master file at a time. The use of the SIGNOFF operator causes the actual deletion of the subfile library. Once SUBFILE has been specified, any query which does not contain a SUBFILE operator causes the query following to be directed against the same input source as that of the preceding query, unless a FILE operator is used to specify otherwise.

The format of the SUBFILE operator is:

```
SUBFILE [subfilename]
```

a. The subfilename is the user-assigned name for the subfile to be created. It must conform to NIPS naming conventions in that it must begin with an alpha character, must not contain any special characters, and must be seven bytes or less. It must not have been previously assigned to any subfile which currently exists.

b. The absence of a user-assigned name requires a unique internally generated system name to be assigned to the subfile. This name is displayed at the terminal prior to the generation of the output report and is available for use with a FILE operator in a later query.

The format of a system assigned name is:

```
Shhmmll
```

where

```
S  = literal value S
hh = hour of subfile creation time
mm = minute of subfile creation time
ll = subfile sequence number
```

The following conventions are applicable in processing of the SUBFILE operator:

a. No more than one SUBFILE operator may be included in a query.

b. Any query which is to create a subfile must contain a KEYWORD, LIMIT, FIND or retrieval IF operator to identify the records which are to be included in the subfile.

c. The use of the SORT operator in a query which creates a subfile does not affect the sequence of the record key entries in the subfile. The record key entries remain in the sequence of the original data file.

d. Any query which requests creation of a subfile but has a syntax error or any other error which prevents the successful completion of the query results in no subfile output. When possible, QUIP is restored to its status prior to the start of the query in error. This allows continuation without adverse effects as a result of the error.

e. In the event of a TP control program error, an operating system failure, a machine malfunction, or any type of error which requires a restart of the TP control program, restoration of the user's subfile library is possible by including the name of the subfile library as an operand of the SIGNON operator, providing the failure was of such a nature that retention of the data set was possible. The first occurrence of the SUBFILE operator causes the naming and allocation of a partitioned data set (library) on the direct access device allocated by the SUBFILE data definition statement in the TP job step. The name of the library is generated by QUIP according to system naming conventions for temporary data sets. This name is displayed at the terminal at the time the data set is allocated and should be noted for possible use later with the SIGNON operator in the case of a restart recovery (see section 3.5.14.1, SIGNON Operator) or the TRACE operator (see section 3.5.14.4, Trace Operator).

f. Use of the SUBFILE operator causes QUIP to be placed in signon mode. This is accomplished at the time the operator is recognized in the input stream, and signon mode remains in effect in the event the query does not successfully complete.

g. Names assigned to subfiles are not available for assignment to other subfiles until the subfile partitioned data set is deleted by means of the SIGNOFF operator (see section 3.5.14.3, SIGNOFF Operator). An error message is issued if a previously assigned name is requested to be assigned to a newly created subfile.

h. A subfile may not be used as a secondary file with Interfile Output. It may be used as the primary file, however.

Example:

```
FILE TEST360. CLASS UNCLASSIFIED
IF SERV ## EQ N,W.
SUBFILE NAVARMY
LIST UIC ALL MEQPT ALL MEPSD
```

Comments: If this is the first request for a subfile of the TEST360 file, a partitioned data set is named and allocated as a result of this query. A member of that partitioned data set is created with the name NAVARMY. That member consists of the record keys of those records in the TEST360 file where SERV is equal to N or W (the file service codes for Navy and Army). An output report would be displayed which listed the UIC, MEQPT and MEPSD fields for those records.

Suppose the next query entered is:

```
IF MEPSD GT 10.
LIST UIC MEQPT
```

Since the previous query created a subfile, NAVARMY, and no FILE operator is included in this query, the input source for the query is the subfile NAVARMY. The query examines those records of the TEST360 file whose record keys are in

NAVARMY (those where SERV equals N or W), qualifies those whose MEPSD value is greater than 10, and lists the UIC and MEQPT fields for those records.

Suppose the next query entered is:

```
IF SERV ## EQ N AND MEPSD GT 10.
SUM MEPSD FINAL PRINT SUM
SUBFILE NAVY
```

Since this query does not include a FILE operator and the previous query did not create a subfile, the input source is the same as that of the previous query, namely the subfile NAVARMY. This query adds a new member to the subfile library with the member name NAVY. It consists of record keys for those records of TEST360 where SERV is equal to N and MEPSD is greater than 10. The output report consists of a total of the MEPSD values for those records.

Suppose the next query entered is:

```
FIND CNTRY NE FRANCE.
SUBFILE
```

The input source of this query is the subfile NAVY created by the previous query. A new member is added to the subfile library with a system generated name which is displayed prior to generation of the output report. The subfile created consists of record keys for the records of TEST360 whose record keys are in subfile NAVY and have a value in the CNTRY field which is not FRANCE. The output report consists of a count of the records that qualify.

Suppose the next query entered is:

```
FILE NAVARMY.
KEYWORD COMMENT INCLUDES ASSIGNED, ALLOCATED.
SUBFILE GT10
LIST UIC COMMENT
IF MEPSD GT 10.
```

In this query a FILE operator is used to refer back to the subfile NAVARMY as the input source. A candidate list is generated as a result of the KEYWORD statement, and the

TEST360 records for those candidates which are also entries in the NAVARMY subfile are examined for MEPSD values greater than ten. The subfile GT10 is created consisting of the record keys of those records which qualify. The output report lists the contents of the UIC and COMMENT fields from those records.

Finally, suppose the following query is entered:

```
FILE TESTABC.
IF FLDA EQ &FLDA.
LIST FLDA
```

Since the FILE operator references a file that is not the master file in this interrogation sequence (TEST360) nor a subfile (of TEST360), the subfile library created by the previous queries is deleted, if the query is free from translator errors. If errors are found, the status of QUIP is restored to that at the end of the last successful query.

3.5.14.3 SIGNOFF Operator

The SIGNOFF operator allows QUIP to terminate, in the online mode, when a SIGNON or a SUBFILE operator has been used in a previous query. It may appear by itself without any other operators. If it appears in a query with other operators, QUIP termination is accomplished after the query in which the SIGNOFF operator appears is completed.

The effect of the SIGNOFF operator is to cancel a previous SIGNON or SUBFILE operator(s) and remove QUIP from signon mode. Any subfiles which may have been created are deleted. Use of this operator requires any subsequent query to include a FILE operator to identify the source of input.

The format of the SIGNOFF operator is:

```
SIGNOFF
```

3.5.14.4 TRACE Operator

The TRACE function provides the user with the capability of reviewing any existing subfile library or any member of

such a library. The operator must appear by itself, or it will be ignored or treated as an unidentifiable operand of another operator.

Subfiles are discussed in section 3.5.14.2. Subfiles are stored as members of a subfile library (PDS) generated for that purpose. The time of creation of each subfile is maintained to permit review in the chronological order in which they were created. In addition, hierarchical (family tree) data is generated to permit the logical review of the hierarchy of the subfiles without regard to the order in which they were built.

Options exist which permit the user to view one specific subfile, with or without record keys, or all subfiles, in chronological or in logical sequence, without record keys.

The following data is displayed for each subfile: the run identifier, subfile name, number of record keys (query hits), level (logical) and sequence (chronological) numbers, the sequence number of the preceding subfile in logical sequence, the lowest and highest record keys, the query used to create the subfile. If the record keys exceed 80 characters they will be truncated to a length of 80 characters for display.

In addition, if a specific subfile is being traced, all record keys associated with the subfile may optionally be displayed.

The format of the TRACE operator is:

```
TRACE  [ACTIVE                ]  [FORMAT                  ]
       [                      ]  [no entry                ]
       [subfile library name  ]  [subfile name[ RECID ]   ]
```

a. The ACTIVE operand specifies that the currently active subfile library is the input to TRACE. ACTIVE is the default in the online mode and may be omitted. In the batch mode ACTIVE is the invalid operand.

b. Subfilelibraryname permits tracing of any existing subfile library. This is a required operand in the

batch mode. Further, a SUBFILE data definition statement must be added to the QUIP step. Generation of subfile library name is discussed under SUBFILE conventions in section 3.5.14.2.

c. The FORMAT operand specifies that statistical data for all files in the subfile library will be displayed in hierarchical sequence.

d. If no entry is specified for the second operand, it defaults to a listing of data for all subfiles on the library in chronological order.

e. The subfilename as the second operand limits the trace to the named subfile.

f. The RECID operand specifies that all record keys will be included in the data for the named subfile. RECID is valid only when preceded by a subfilename.

Examples:

a. TRACE

Lists trace data for all subfiles in the currently active library in chronological sequence.

b. TRACE ACTIVE FORMAT

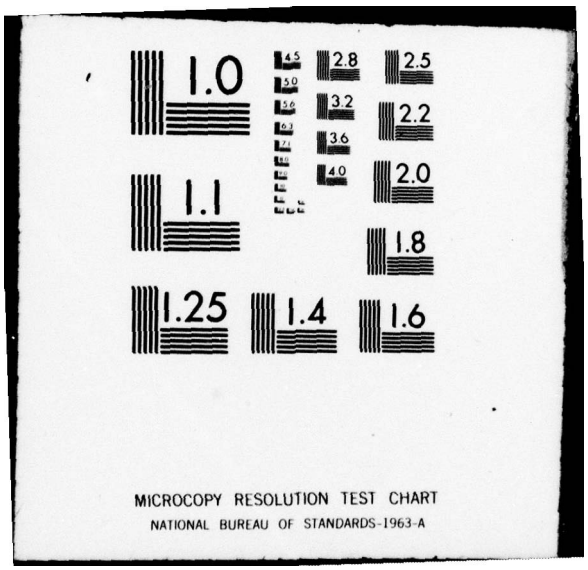
Lists trace data for all subfiles in the currently active library in hierarchical sequence.

c. TRACE NMCSSC1.S132205 FORMAT

Where NMCSSC1.S132205 is a subfile library name. This would provide a list of trace data for all subfiles in the named library in hierarchical sequence. In the situation where the TP program abnormally terminated or an operating system or hardware failure occurred, the system may be able to retain subfile libraries. If TP is restarted, existing subfile libraries could be traced online. If not, the TRACE could be submitted in a batch QUIP job. Addition of a SUBFILE data definition

AD-A063 431 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 9/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78 C K HILL
UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-6 SBIE-AD-E100 131 NL

3 OF 4
AD
A063431


1.0
2.8
2.5
3.2
2.2
3.6
2.0
4.0
1.1
1.8
1.25
1.4
1.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

statement is required when using the TRACE operator in the batch mode.

d. TRACE S132403 RECID

Where S132403 is the system generated name of the third subfile created on the currently active subfile library. This would produce the trace data for the named subfile. All record keys will be listed.

3.5.14.5 FORMAT Operator

The user may pass from one to five format names from his QUIP query to FORMATTER. To accomplish this, the QUIP query or Load RIT must generate an output line that contains the characters, 'FORMAT=', in the first seven positions. The complete form of the line is as follows (with no intervening spaces):

```
FORMAT = name
         (name1,name2..name5)
```

The order in which the format names are provided, determines the sequence in which they are processed by FORMATTER. Format name1 is processed first and format name5 is processed last. If previous queries or logic statements have passed format names to FORMATTER, the names provided by the present query are stacked at the end of the list. The FORMATTER format name list may contain a maximum of five names at a time. As each format is processed, the name is removed from the top of the list.

The capability to dymamically specify formats in QUIP is most useful when FORMATTER recycles. Normally, because QUIP invokes PAGE after execution, FORMATTER would not recycle. By using two QUIP operands in his query, the user can achieve automatic recycling when FORMATTER invokes QUIP and also save his query output for later use.

3.5.14.6 VIEW Operator

VIEW = member name

This parameter causes the results of the query to be stored on the specified VIEW data set member. The member may be an existing member, in which case, the member is replaced or it may be a new member, in which case, the member is added. (The QUIP source statements will not appear in the VIEW member.)

3.5.14.7 OMQ Operator

OMQ = YES / NO

This parameter is used to indicate whether the output from the QUIP query is to be placed on the OMQ. The default value is YES. If the user wishes to have the query output placed only on the VIEW data set, he must specify OMQ=NO.

These parameters must appear at the beginning of the query, before any QUIP operators. Note that the VIEW and OMQ parameters are not exclusive elements of dynamic format specification but may be used in any query.

3.6 Field Modifiers

Several special field modifiers exist to override FFT parameters or normal modes of operation such as ALL, ##, '', and m/n.

3.6.1 Periodic Set Control (ALL)

The ALL modifier may precede any field name used with a COMPUTE, COUNT, DISPLAY, EXTRACT, LIST, or SUM operator allowing processing of all periodic subsets rather than just the flagged ones. Since the logical use of the modifier pertains only to periodic data, it is an effective "no operation" if applied to fixed set data and has no effect on system performance. Since a Source Direct query without a

FIND operator or a retrieval type IF operator will process all the periodic subsets, the use of the ALL modifier is redundant in those instances. Its use, however, does not interfere with system functioning.

3.6.2 Subroutine Expressions (##)

The name of a subroutine to be used in converting a field (input or output, depending on the operator) may be enclosed in pound signs (##) and placed after the field name. The subroutine specified in the input stream is used in place of the subroutine specified in the FFT. If the user desires to eliminate subroutine conversion as specified in the FFT, the override feature may be used by following the field name with a double pound sign (##). A subroutine expression may be used with the following operators: DISPLAY, EXTRACT, FIND, FOR/AND, retrieval and output type IF, LIMIT, LIST, and SORT.

3.6.3 EDIT Keyword ('')

The user may supply an edit mask to be used for output or to override the FFT-specified edit mask for a numeric file field. The edit mask to be used is enclosed in apostrophes and placed after the keyword EDIT which is coded after the name of the field or literal to be edited. This keyword may be omitted in the LIST statement. An edit mask specified in the FFT may be eliminated by placing a double apostrophe ('') after the word EDIT (or after the field name in the LIST statement if the word EDIT is omitted). When both an edit mask and subroutine name follow a field name, it makes no difference which is coded first. An edit specification may be used with the LIST and the DISPLAY operators. The user should check the format of these operators to see which operands permit this entry.

3.6.4 Partial Field Notation (m/n)

The user may specify that only part of a field is to be used through partial field notation. The format of this

notation is m/n, where m is the first character position to be used and n is the last character position to be used. Partial field notation may be used with the DISPLAY, EXTRACT, FIND, retrieval and output type IF, LIMIT, and SORT operators, but it may not be used with a binary or coordinate mode field. The user should check the format of each of these operators to see which operands permit this specification.

3.7 Universal Match Character ($)

The use of dollar signs ($) as characters of a literal which is not enclosed in apostrophes provides universal match capabilities for those characters of the literal. The appearance of a dollar sign causes the character to be ignored when it is compared to its corresponding character in the file field; or, in other words, the dollar sign may be interpreted to represent any data value character which is unknown or considered unimportant and is to be ignored. Literals are normally padded with blanks on the right, if alphabetic, and with zeros on the left, if decimal. Use of the dollar sign is adjusted to allow for this logical padding. For example, use of the dollar sign as a trailing character on an alphabetic field causes the literal to be padded on the right with dollar signs. Likewise, use of the dollar sign as a leading character on a decimal field causes the literal to be padded on the left with leading dollar signs. Alpha literals may contain embedded dollar signs. Embedded dollar signs may not appear in decimal literals. The dollar sign may not be used against a binary or coordinate field. Subroutine processing is not permitted on a literal which includes the dollar sign unless the literal is enclosed in apostrophes. All references to subroutines will be ignored when a dollar sign is encountered in any literal which is not enclosed in apostrophes. The Universal Match Character may be used with the FOR/AND, the retrieval type IF, and the FIND operators.

3.8 Skeleton Query

A skeleton query is defined by prefixing at least one operand in the query with an underscore character (0-5-8

punch). The presence of this character automatically causes that query to be generated so that the designated operands can be replaced by other operands for succeeding executions. The operands cannot be replaced during the same run in which the query is being compiled. Operands which are legal replaceable variables are fieldnames, relational operators, and literals. Replacement variables may be specified on the LOAD QUERY operator to replace the replaceable operands.

3.9 Noise Words

The following is a list of noise words. These words are ignored in the input stream. They should not be used as field names or literals, unless they are enclosed in apostrophes (5-8 punch).

```
ARE
FIELD
FIELDS
IS
OF
RECORDS
THAN
THAT
THE
TO
, (0-8-3)
```

3.10 Operator Initiators

The following words and special characters may be used to separate (or initiate) operators. They will not cause statement termination and may not be used where a statement terminator is not allowed:

```
THEN
; (11-8-6)
```

Example

```
IF PERS GT 0 THEN COMPUTE AVERAGE OF PERS.
```

3.11 Logical Connectors

The following words and special characters may be used as logical connectors:

```
OR
        (12-8-7)   (OR)
AND
&       (12)    (AND)
```

3.12 Relational Operators

The following words and special characters may be used as relational operators to describe a relation between a field and a literal(s) or field:

```
EQ
EQUAL
EQUALING
EQUALS
=   (8-6)    (EQUAL)
LESS
  <   (12-8-4)    (LESS)
GREATER
  >   (4-8-6)     (GREATER)
BETWEEN
CHANGES
GT          (GREATER)
GE          (GREATER OR EQUAL)
LT          (LESS)
LE          (LESS OR EQUAL)
BT          (BETWEEN)
NE          (NOT EQUAL)
ABSENT
CONTAINS
```

The modifier NOT may be used in front of all the relational operators to reverse the meaning; i.e., "NOT EQ" means NOT EQUAL, "NOT NE" means EQUAL, except that NOT may not precede EQ or BT in a LIMIT statement and NOT may not precede CHANGES in the output type IF statement.

3.13 Geographic Operators

Two geographic search operators are provided:

a. OVP, OVERLAP, OVERLAPS (tests if a file point, line, or polygon overlaps, intersects, or is contained within a literal which is defined as a point, line, or polygon.)

b. CIR, CIRCLE (tests if file geographic point lies within a circle which is defined as a point and radius)

3.14 Query Documentation

3.14.1 NOTE Operator

The NOTE operator may be used to insert comments into the query. The NOTE statement is terminated by a period or the end of the query. Any words or characters may be included within the NOTE statement.

3.15 Sample QUIP Queries

Three QUIP queries are presented in this section. The first is a relatively complex query which uses virtually all of the QUIP operators. The second is a sample of a QUIP/RIT query which uses a stored RIT to generate the output display. Immediately below the source statements for each query is a narrative explanation of the request. The third is a sample of a QUIP query which uses IFO.

The job setup for QUIP execution as a background job, including instructions for storing queries on a user library, is described in NIPS 360 FFS, Vol. VIII, Job Preparation.

SAMPLE QUERY1

```
FILE SOFAA. CLASS UNCLASSIFIED                    1
IF SERV EQUAL ARMY NAVY MARINE USAF.              2
SORT SERV.                                        3
```

```
IF SORTKEY CHANGES PRINT SUM2 COUNT2 EJECT                          4
        INITIAL DISPLAY 'SORTKEY = ' SORTKEY.                       5
LIST UNAME COMDR PERS ALL MEQPT                                     6
FOR SERV = ARMY NAVY MARINE USAF SUM1 PERS COUNT1 UNAME
          ALL MEQPT.                                                7
COUNT2 UNAME ALL MEQPT SUM2 PERS                                    8
COMPUTE PERSAV EQUAL AVERAGE PERS                                   9
FINAL PRINT SUM2 COUNT2 EJECT                                      10
PRINT SUM1 COUNT1                                                  11
SPACE 3                                                            12
DISPLAY 'AVERAGE PER/UNIT = ' PERSAV                               13
HEADER1 'QUIP TEST RUN'                                            14
HEADER2 'NUMBER 1'                                                 15
TRAILER 'END OF PAGE'                                              16
```

The numbers in the following discussion of the sample QUIP query apply to the input line being referenced and correspond to the number on the right side of the source listing.

Line 1 defines which file is to be used as a data source and its classification. These two operators must appear in every run. The FILE operator is terminated by the period after SOFAA.

The retrieval parameters are defined in line 2. Only records in which the field SERV is equal to ARMY, NAVY, MARINE, or USAF will be presented to the output portion of the query. The retrieval IF is terminated by the period after USAF.

The data records will be sorted on the field SERV before being output (line 3).

Lines 4 and 5 are an output type IF operator. The output operators in the range (to the next period) of this IF will be executed only when the condition is true. In this example the objective is to print the sum and count work areas, eject to a new page, and display the new sort key whenever the sort key changes. The first sort key is displayed by the use of the INITIAL operator. The first time the condition is tested it cannot change because there is no old value to compare against; therefore control is

passed to the INITIAL operator or to the end of the statement if INITIAL is not present.

Line 6 is a simple list operator requesting the listing of three fixed set fields and all the MEQPT fields. MEQPT is a periodic field and will not be flagged by the retrieval (line 2); therefore, the ALL modifier must be used to output unflagged subsets.

Line 7 contains a SUM of the field PERS and a COUNT of a fixed and a periodic field. These operators are conditioned by the FOR operator so that they will create different work areas depending on the value of the field SERV.

Line 8 contains additional SUM and COUNT operators. These are not conditioned by the FOR operator because of the period at the end of line 7. It is the intent in this example to create overall totals to be output at the end of the run and every time the sort key changes.

To compute the average of the PERS field, the COMPUTE operator is used in line 9. This average will be output by a DISPLAY operator later in the run (line 13).

Line 10 begins with the FINAL operator which marks the end of the "output loop" of QUIP. The operators between the start of the query and the FINAL operator will be executed, in order, for every record which passes the qualification parameters (retrieval type IF (line 2), or LIMIT). All operators after the FINAL operator will be executed only when an end-of-file is received from the data base. The operator will then print out subtotals SUM2 and COUNT2 for the last sort key, eject to a new page, and print out a grand total matrix (line 11). Three spaces are then skipped and the average of the PERS field is displayed.

Lines 14 and 15 are headers and will appear at the top of every page of output in the order they appear in the source. Line 16 is a trailer and will appear at the end of every page of output.

SAMPLE QUERY2

```
FILE TEST360. CLASS UNCLASSIFIED                          1
IF SERV EQUAL ARMY NAVY MARINE USAF.                      2
SORT SERV.                                                3
LOAD RIT 'QPRIT' PARAM='ARMY'.                            4
```

Line 1 defines the file to be used and the classification of the output.

Line 2 is a retrieval type IF statement.

Line 3 defines the field which the qualifying records are to be sorted on prior to output execution.

Line 4 identifies the RIT to be used to format the output of those records satisfying the conditions in the query.

The following sample query uses two secondary files, TESTA and TESTB, to illustrate the use of IFO. File fields prefixed by A and B are from TESTA and TESTB, respectively.

SAMPLE QUERY 3

```
FILE TEST360.  CLASS UNCLASSIFIED                         1
DISPLAY UIC                                               2
IF UIC NE AA0000                                          3
LIST MEQPT                                                4
EXTRACT MEQPT 1/1 TESTA.                                  5
DISPLAY AMEPSD 10 AMEORN 10 AMEORC 20 AMEQPT              6
IF AMEORN = 0                                             7
COMPUTE TOTMPSD = TOTMPSD + AMEPSD.                       8
EXTRACT SERV ## TESTB                                     9
LIST BUIC                                                10
COUNT BUIC                                               11
EXTRACT                                                  12
DISPLAY 'TOTAL POSS.' 4 TOTMPSD PRINT COUNT              13
COMPUTE AVER = AVERAGE MEPSD                             14
DISPLAY AVER                                             15
```

Line 1 defines the primary file and its classification.

Line 2 outputs the field UIC from TEST360.

Lines 3, 4 and 5 are an output type IF statements with the output operators LIST and EXTRACT. These operators are executed only when the condition UIC NE AA0000 is true. Since lines 6, 7, and 8 are in the range of the EXTRACT operator in line 5, if the condition in line 3 is not true, processing continues with line 9 following the range of the EXTRACT operator.

The EXTRACT operator in line 5 initiates the IFO process by accessing all the records in the secondary file TESTA which have the first character of their record ID equal to the first character of MEQPT in the first subset of Set 1 in the current record from TEST360. After the specifications in lines 6, 7, and 8 are processed against each of those records, the first character of MEQPT in the next subset is used to access the corresponding records in TESTA which are also processed by lines 6, 7, and 8. This process continues until the set in the current TEST360 record is exhausted.

Line 6 is used to output various fields from the TESTA secondary file record retrieved by the EXTRACT operator in line 5.

Lines 7 and 8 are an output type IF statement with the output operator COMPUTE executed only if the condition in line 7 is true. The running total of the secondary file field AMEPSD is computed for each record retrieved by the EXTRACT operator in line 5. The total is later output by a DISPLAY operator in line 13.

Line 9 contains another EXTRACT operator. It terminates the range of the first EXTRACT operator in line 5 and initiates another IFO process by accessing all the records in the secondary file TESTB which have the first character of their record ID equal to SERV (without being converted by the FPT specified subroutine) in the current primary file record.

Lines 10 and 11 process the TESTB records accessed by the EXTRACT operator in line 9.

Line 12 contains an EXTRACT operator without operands which signals the end of the range of the EXTRACT operator in line 9 and the return to primary file (TEST360) processing.

Line 13 outputs the COMPUTE result from line 8 and the COUNT result from line 11.

Line 14 computes the average of the TEST360 field MEPSD.

Line 15 outputs the COMPUTE result from line 14.

3.16 Operator Summary

This operator summary is provided in alphabetic sequence as a quick reference for users already familiar with the QUIP language. It contains a brief description and entry format for each operator.

CLASS Operator

The CLASS operator is used to define the classification of the output report.

CLASS classification literal [N]

COMPUTE Operator

The COMPUTE operator is used to perform the arithmetic operations of addition, subtraction, multiplication, and division; it is also used to calculate the statistical functions of average, percent, and variance. This operator is associated with three format-types, the choice of which is determined by the function being performed.

COMPUTE user-name [xx] EQ { [ALL] fieldname / literal } operation

[ALL] { fieldname / literal } operation... { [ALL] fieldname / literal }

COMPUTE user-name EQ function fieldname(s)

COMPUTE user-name

COUNT Operator

The COUNT operator, which may be followed by one or more alpha or numeric field names, is used to count the occurrence of records which contain the specified fields.

COUNTn [ALL] fieldname... [ALL] fieldname

DISPLAY Operator

The DISPLAY operator is used to display literals and/or the value of file fields, sort key entries, and the user-named result fields from a COMPUTE statement.

```
                  ( 'literal'
                  | SORTKEY        [m/n]
                  |                [EDIT   ''              ]
DISPLAY [nn]      |                [EDIT   'edit mask'     ]
                  | [ALL] field     [#subtab#         [*mm] ]
                  |                [##                     ]
                  ([ALL] (field [nn] field [nn]...field[pp] *mm)
```

EJECT Operator

The EJECT operator is used with the paging mode if output to terminate the current page and initiate the next.

EJECT

EXTRACT Operator

The EXTRACT operator is used to initiate the IFO process by signalling the interruption of primary file processing and the retrieval from a secondary file of one or more data records, the selection of which is based on the contents of the pointer field used with the operator.

```
                  [ ( [ALL]   IFOGRPx                          ) ]
                  [ (                                          ) ]
EXTRACT           [ ( [ALL]   fieldname  [##          ]        ) ]  filename
                  [ (                    [#subname#   ]        ) ]
                  [ (                    [m/n         ]        ) ]
                  [ ( SORTKEY [m/n]                            ) ]
                  [ ( literal name                             ) ]
```

FILE Operator

The FILE operator is used to identify the file to be queried. When IFO is used it identifies the primary file. The operator has two formats, the choice of which is dependent on the input source. If the file name is a qualified data set name, it must be enclosed in apostrophes.

```
{FILE } {filename.   }
{QUERY} {subfilename}

{FILE } filename retrieval-number [RITname].
{QUERY}
```

FINAL Operator

The FINAL operator conditions all of the operators which follow it so that they will be executed only after an end-of-file condition has been sensed on the data base.

```
        { COMPUTE }
        { DISPLAY }
        { EJECT   }
FINAL   { MARK    }
        { PRINT   }
        { SPACE   }
```

FIND Operator

The FIND operator is used to specify the number of qualifying records to be retrieved before end-of-file is reported.

```
                                  {fieldname}                                {geo. op.}
FIND  [n] [ANY] {groupname} [m/n][#subtab#][NOT] {rel. op.} literal(s)

                                conditional clause

                          ... {AND} conditional clause  .
                              {OR }
```

FOR Operator

The FOR operator is used to generate a one-dimensional matrix. Another dimension may be added to the matrix when this operator is used in conjunction with the AND logical connector.

```
                    [  ##    ]
FOR fieldname       [#subtab#]  relational
                                operator    literal(s)

                    [  ##    ]
AND fieldname       [#subtab#]  relational
                                operator    literal(s)

{SUMn [ALL] fieldname[m] -- [ALL] fieldname [m] [VLONG]} [HTOTAL]
{COUNTn[ALL]fieldname --- [ALL] fieldname

{SUMn[ALL]fieldname[m] --- [ALL]fieldname[m][VLONG]} [HTOTAL]
{COUNTn[ALL]fieldname ---[ALL]fieldname
```

HEADER Operator

The HEADER operator is used to place descriptive information at the top of every page of output.

```
HEADERn  literal
```

IF Operator (output type)

The output type IF operator allows the output operator(s) within the statement to be executed only if the condition is found to be true.

```
                  { fieldname                }
IF                { groupname [ #subtab# ]   } relational          output
                  {      SORTKEY   [ m/n ]   } operator  [literal] operator(s).
```

conditional clause

IF Operator (retrieval type)

The retrieval type IF operator provides the logical criteria for record selection.

```
IF [ANY] { fieldname } [m/n] [#subtab#] [NOT] { geo. op } literal(s)
         { groupname }                        { rel. op }
```

conditional clause

```
... { AND } conditional clause .
    { OR  }
```

KEYWORD Operator

One retrieval type KEYWORD operator provides the logical criteria for record selection based upon keyword indexed fields having the qualifying values in the Index Data Set.

```
KEYWORD fieldname INCLUDES [ value(s)   ]
                           [ literal(s) ]
```

LIMIT Operator

The LIMIT operator is used to limit the records read from the file to specified portions of the RECORD ID.

```
LIMIT { fieldname } [m/n] [#subtab#] relational operator literal.
      { groupname }
```

LIST Operator

The LIST operator is used to list the contents of the file fields.

```
                                  [ EDIT     ''              ]
                                  [ EDIT     'edit mask'     ]
LIST [ALL] {fieldname} {groupname}[ ##                       ]  . . .
                                  [  #subtab#                ]
```

LOAD QUERY

The LOAD QUERY operator is used to load a stored query from the user's library and execute it. The operator also permits the user to define replacement variables when executing a stored query.

```
LOAD QUERY   'query-name'  ['library-name']

    [(keyword=replacement variable)].
```

LOAD RIT

The LOAD RIT operator is used to execute a RIT from the user's library to format the output display.

```
LOAD RIT 'ritname'  ['libname']  [PARAM='parm list']
                          [{NO       }]
                  EXTRACT= [{filenames}]  .
```

MARK Operator

The MARK operator is used to print a line of asterisks or to print a literal embedded in the center of a line of asterisks.

```
MARK   [literal]
```

NLINES Operator

The NLINES operator may force the page or the display mode of output. Normally, it forces the paging mode of output and allows the user to specify the number of lines to be printed per page. The default value associated with this operator is 60 lines of output per page.

```
NLINES  [=]  numeric literal
```

PAGENO Operator

The PAGENO operator is used to reposition the page number on each output page. The printing of the page number cannot be suppressed. The default position for the operator is LOWER RIGHT.

```
                 {LOWER}  {RIGHT}
PAGENO           {UPPER}  {LEFT }
```

PRINT Operator

The PRINT operator is used to print the results of previously defined SUM or COUNT operators.

```
                 {SUMn  }       {SUMn  }
  PRINT          {COUNTn} ... n {COUNTn}
```

SIGNOFF Operator

The SIGNOFF operator allows QUIP to terminate in the online mode when QUIP is in the signon mode. If any subfiles exist, it causes them to be deleted.

```
SIGNOFF
```

SIGNON Operator

The SIGNON operator is used in the online environment to place QUIP in the signon mode which indicates that QUIP is to pass FILE and CLASS operator information to succeeding queries. It also is used to identify a subfile library that is to be restored to availability in the event a restart of the TP system is required during a terminal session which has created subfiles.

```
SIGNON [name]
```

SORT Operator

The SORT operator is used to specify the fields which are to be used to resequence the file or the answer records.

```
        {fieldname}                [#D#      ]          {fieldname }          [#D#      ]
SORT    {groupname} [m/n] [ or     ] ... {groupname} [m/n] [ or     ]
                          [#subtab#]                                [#subtab#]
```

SPACE Operator

The SPACE operator is used to generate blank lines in the output.

SPACE [numeric literal]

SUBFILE Operator

The SUBFILE operator provides the online capability to create and process as an input source a subfile consisting of selected entries retrieved from a NIPS file. Its use places QUIP in the signon mode.

SUBFILE [subfilename]

SUM Operator

The SUM operator, which is followed by one or more numeric fields, is used to sum the contents of the specified fields.

SUMn [All] fieldname[m] ... [ALL] fieldname[m]

SYSDATE Operator

The SYSDATE operator is used to reposition the system date (DD MM YY). The system date is printed on every page of output and cannot be suppressed. The default position for this operator is UPPER RIGHT.

```
SYSDATE {UPPER} {LEFT }
        {LOWER} {RIGHT}
```

TRACE Operator

The TRACE operator provides the capability to review statistical data for a specific subfile on a subfile library, with or without record keys, or all subfiles in chronological or hierarchical sequence without record keys.

```
TRACE  [ACTIVE                ]  [FORMAT                    ]
       [                      ]  [no entry                  ]
       [subfile library name  ]  [subfilename [RECID]       ]
```

TRAILER Operator

The TRAILER operator is used to place descriptive information at the bottom of every page of output.

TRAILERn literal

3.17 QUIP Definitions

Statement - A QUIP statement consists of one or more operators ended by a statement terminator. Some operators must appear only as a statement; i.e., the IF operator. The statement determines the range of the operator.

Operator - An operator is a word or a special character which will cause some specified actions to be taken during output.

Statement Terminator - The special character which may be used to terminate a statement; i.e., period(.)

Operator Initiator - The word or special character which may be used to terminate one operator and initiate the next; i.e., THEN, semicolon (;).

Literal - A string of characters supplied by the user for some special purpose; i.e., edit mask, classification, comparison value, etc. Literals must be delimited by apostrophes (8-5 punch) unless otherwise

stated in the operator descriptions. Any character may appear in a literal which is delimited by apostrophes except another apostrophe.

Self-defining literal - An alphameric character string containing no blanks or special characters other than a dash (-) or leading or trailing dollar signs ($). A self-defining literal may or may not be delineated by apostrophes at the user's option.

A numeric literal - A digit, or string of digits. A numeric literal must not be enclosed in apostrophes (8-5 punch). A numeric literal may be signed (±) or unsigned (considered positive.) The sign may be prefixed to the literal or may appear as zones to the units positions.

Indirect address literal - The name of a field, the contents of which may be used as a data value. The field name must be preceded by an ampersand (&).

Logical Connector - A logical connector is used to logically connect two or more conditions in an IF statement; i.e., and, &, or, .

Relational Operator - A relational operator is a word, group of words, or a special character which describes the relation between a field and another field or a literal which is to be tested.

Field Name - The normal system field name as specified in the FFT for the file. (A field name must be seven characters or less.)

Geographic Operator - One of the system-defined geographic operators; i.e., OVP, CIR.

Skeleton Query	- A stored query in which words preceded by an underscore character may be replaced at execution time.
Interfile Output (IFO)	- IFO is the capability to combine data elements from several related files during the output process.
Primary File	- In IFO, the primary file is the file entered with the FILE operator. It contains data elements which identify records in other referenced or secondary files.
Secondary File	- In IFO, a secondary file is identified by the final operand on an EXTRACT statement. The selection of records from a secondary file depends on the contents of a pointer field in the primary file specified on the EXTRACT statement.
Pointer	- A pointer is a data element (control, fixed, or periodic) in the primary file which is used to identify records in referenced secondary files. (It consists of the record key or the high-order portion thereof for a secondary file but may also include a secondary file identifier. It may also be the sort key or a COMPUTE result.
Range of an EXTRACT Operator	- The range of an EXTRACT operator describes the source statements (coded following an EXTRACT statement) which contain the logical specifications for records retrieved from the secondary file by the preceding EXTRACT operator. The range is terminated by another EXTRACT operator by the end of the query.
Index Data Set	The data set containing all information required to support the indexing capability. It contains the secondary and keyword indexed fields along with their associated data values and record IDs.

Signon Mode	The status of QUIP in the online environment from the time the SIGNON or SUBFILE operator is recognized until a query which contains the SIGNOFF operator successfully completes. While QUIP is in the signon mode, any query which does not include a FILE operator uses the FILE and CLASS operator information of the previous query, providing the previous query did not create a subfile. If a subfile was created in the previous query and the FILE operator has not been specified since, the subfile created by the previous query is used as the source of input. Therefore, while QUIP is in the signon mode, the FILE operator is required only for the first interrogation of a file or when a change of input source is required.
Subfile	A file consisting of selected record key entries which are selected based on conditional expressions in retrieval and/or limit logic of a query which contains a SUBFILE operator. The subfile is stored as a member of partitioned data set which is dynamically allocated on system direct access work space.
Master File	The NIPS FFS data file from which subfiles are created, selected record keys of which comprise the subfiles.
Terminal Session	A series of terminal activities initiated by a LOGON command and terminated by a LOGOFF command.

Section 4

SOURCE DATA AUTOMATION (SODA)

The Source Data Processor provides the user with a means of executing a precompiled File Maintenance logic statement against an update transaction entered by an operator at an on-line terminal. The same transaction could also be punched on cards and processed by the batch-mode File Maintenance (FM) processor. Given a file, SODA will retrieve the record identified in the update transaction. It will then provide this record, and the transaction, to the library logic statement identified in the update transaction. After the update, SODA will hold the record temporarily and eventually return it to the master file in updated form. The user's input transaction is validated by the library logic statement during execution. If errors are detected, immediate notification will be given to the terminal operator. The user may correct individual lines of his transaction, using the standard TP correction procedures (see subsection 2.3.3.3) and repeat SODA request. He may alternatively enter Key-numbered corrections to individual fields and repeat the SODA request.

NOTE: SODA cannot be used to process a VSAM file.

4.1 Language

A keyword language is used for input to SODA and consists of control statements identifying files, records, report formats, corrections and disposition actions. It is used in addition to the standard TP language described in section 2.

4.1.1 Input Line Formats

Keyword statements are free-format. They may begin on any part of a line, and they may have any number of blank spaces between words in the statement. A line is a single physical console transmission. On the local 2260, it begins

with the START symbol, ends with the ENTER symbol, and it contains, at most, 80 characters between the two symbols. On other devices, a similar line definition can be made. Multiple lines can be used to submit the data for one execution of SODA. A word or literal must, however, appear entirely in a single transmission, and it cannot be split between two transmissions. Statement terminators (.Ø) are optional for all statements.

In contrast with keyword statements, update transactions are not free-format and normally begin with the first position of a line. All subsequent blanks and characters are included in the transaction. SODA will also accept update transactions that begin in the same line as a Keyword statement. In that case, the first nonblank character following the keyword statement is assigned to column one of the transaction.

4.1.2 Keywords

The keyword language includes the following control operators:

a. Normal Operation

- o FILE
- o REPORT
- o CANCEL
- o UPDATE

b. Correction Mode

- o Keynumber
- o INSERT
- o CHANGE
- o DELETE.

4.1.2.1 FILE

The FILE keyword is followed by a word that indicates which data set shall be updated, e.g., FILE TEST360. Only disk index sequential files can be processed by SODA. The file name may be a qualified data set name, in which case the NIPS file name would be the last segment of the fully

qualified data set name. Because of the embedded special characters, a qualified data set name must be enclosed in apostrophes.

4.1.2.2 REPORT

The REPORT keyword is followed by a word that identifies the report used with the update transaction, e.g., REPORT REMOTE. Library logic statements are organized by file-report-statement. The word following the REPORT operator will be used by SODA as the second-level key in the search for the logic statement (the word following the FILE operator is the first level, and up to six characters from the update transaction are used as the third level).

4.1.2.3 CANCEL

The CANCEL keyword tells SODA to delete a previously updated record from the temporary hold file so that it will not be returned to the master file. Two formats are allowed:

```
CANCEL ALL

CANCEL record key,.....,record key
```

The first deletes all entries from the hold file. The second identifies one or more specific records on the hold file.

4.1.2.4 UPDATE

The UPDATE keyword tells SODA to copy a previously updated record from the temporary hold file back to the master file, and then to delete that record from the hold file. As with CANCEL, two formats are allowed:

```
UPDATE ALL

UPDATE record key,...,record key
```

The first causes transfer and deletion of all records on the hold file. The second, with one or more record keys, causes transfer and deletion of only the specified records. When

the user has entered all transactions for a file and wishes to transfer these to the master file, he must sign off with UPDATE ALL.

4.1.3 Value Words

Following any keyword, there must be a word assigning a value; e.g., FILE TEST360 assigns a data set name, and CANCEL N52593 assigns a record Key. A value word may be any length appropriate to the keyword it follows. It may not exceed the line size of the console and must not be split between two transmissions. It may, however, appear in a different transmission from the keyword it follows. A Keyword is delimited by a blank or = character (on the local 2260, NEWLINE is treated as a blank). The value word begins with the first nonblank, non-"=" character. The value word may then contain any character except blank or comma, and is ended by either of them, or by the end of the line. A period is not included in the value word if it is the last character of the word. If the user needs to enter a value word that contains a blank, comma, or final period, he may enclose the word in single quotation marks, making a literal; e.g., CANCEL 'KZY##72,93.'. If the literal must also contain the single quote character, type two such characters for each single quote desired; e.g., 'TOM"S#BOOK'.

4.1.4 Elements of an Input Request

A console user enters five types of information: FILE, REPORT, an update transaction, execute SODA, and either UPDATE or CANCEL. The last operators, UPDATE or CANCEL, may not be entered until at least one record has been placed on the temporary hold file. A sixth type, correction data, may be entered at the user's option. (see section 4.3).

4.1.4.1 File Modification Element

A file modification is defined by a FILE operator, a REPORT operator, and the update transaction data in the fixed format expected by the library logic statement. Since normal terminal usage will be for a series of updates to the same file in the same report format, SODA will retain the file and report names on disk. The most recent name from

the terminal will be used until another is entered. Thus the FILE and REPORT operators must be used with the first update transaction, but they are optional on succeeding transactions. At a given terminal, the FILE operator for a second master file may not be entered until the user has signed off on the first master file (see section 4.1.2.4).

4.1.4.2 Execution Request Element

Execute SODA is a control statement to the TP Monitor. On the terminal, it consists of the (logical NOT) symbol followed by the characters FM, or the character F.

(logical NOT)F

It may be entered as a separate line transmission, or entered with the update transaction. If the latter, it will first be removed from the transaction by the TP Monitor, and then the transaction line will be extended with blanks to the defined line size for that console.

4.1.4.3 Disposition Element

CANCEL or UPDATE should be used after multiple updated records have accumulated on the hold file, and must be used after all transactions for a file have been processed. They may optionally be used at any time. A disposition operator is not entered in combination with update transactions or other operators; it appears as a single statement followed by the SODA execution request.

4.2 Input Using a Local 2260

The following are examples of input when using a local 2260 Display Station.

Example 1

(SMI) FILE TEST360 REPORT REMOTE(ENTER) (NEWLINE)

(SMI) AN00147AAB5 (ENTER) (NEWLINE)

(SMI) (logical NOT)FM (ENTER)

```
INPUT SCRATCHED. READY FOR MORE.

---the above message would be written by SODA after
   successful processing of the update transaction---

(SMI) UPDATE NØØ147 (logical NOT) FM (ENTER)
```

Three lines are used to enter the request. The second line is an update transaction. It is a fixed field, as defined for the logic statement "A" in the "REMOTE" report for TEST360. That definition would declare column 1 to be the statement identifier, and columns 2-7 to be the record Key. The fourth line, starting INPUT, would be displayed by SODA to indicate successful processing of the update transaction.

Example 2

```
(SMI) FILE TEST360 REPORT REMOTE (NEWLINE)

CAAB5NØØ147 (logical NOT) FM (ENTER)

INPUT SCRATCHED. READY FOR MORE

---previous line output by SODA---

(SMI) CAAX2N52591 (logical NOT) F (ENTER)

INPUT SCRATCHED. READY FOR MORE

---previous line output by SODA---

(SMI) CBYA3NØ355Ø (logical NOT) FM (ENTER)

INPUT SCRATCHED. READY FOR MORE.

--previous line output by SODA---

(SMI) CANCEL N52591 (logical NOT) FM (ENTER)

CHANGES TO RECORD N52591 CANCELLED.

ØØ2 REMAIN
```

```
---previous two lines line output by SODA---

(SMI) UPDATE (logical NOT)F (ENTER)

ALL UPDATED RECORDS WRITTEN ON TEST360

---previous line output by SODA---
```

In this example, three update transactions are entered, in a slightly different format, for logic statement C in the REMOTE report for TEST360. The execute SODA statement is appended to the last input line and the abbreviated form (logical NOT)F has been used. The first transmission includes FILE, REPORT, and the first update transaction. After the three updates, the user decides to delete his second update and then transfer the remaining two updates back to the master TEST360. More update transactions could follow for TEST360, and the FILE operator would be unnecessary. The hold file is empty, and thus a second master file could be named.

4.3 Corrections to Terminal Transmissions

The following two methods are available to the user to alter his input and retry an unsuccessful update.

4.3.1 Character Substitution Capability

Errors may be detected by the library logic statement during execution. These might include invalid numeric field, replacement field does not verify present contents, or subset not found. The logic statement can flag up to nine transaction fields to be corrected. SODA assigns each field a Keynumber from 1 to 9 in order of detection. It will display the Keynumber as three digits along with the explanatory message provided by the logic statement. Upon completion of the logic statement, the complete transaction will be displayed, and each error field will be underlined by its Keynumber, character-for-character.

```
111 NOT NUMERIC          Messages from
222 SUBSET NOT FOUND     ERR Statements during
                         execution of logic statement.
```

```
---TEST360---REMOTE---

AN00147XYZ2234ABC00K97     Output from SODA at end of
     222       11111       logic statement.
```

The original input is retained and may, of course, be corrected by the usual TP techniques (described in sections 2.3.3.3 and 4.3.2). The following entry, however, can be processed by SODA:

(SMI) 111=00597, 222=XBA (logical NOT) FM (ENTER)

This line, or multiple lines, will be added to the end of the original input by the TP Monitor. SODA will locate the boundary between the update transaction and the correction Keynumber entries, and having retained the column numbers that were underlined will perform the substitution. An entire field should be replaced; but if the underlined field length and substitution field length are not equal, substitution will begin at the first underlining position with the first substitution character and continue until either field is exhausted.

If the processing of a corrected transaction again produces errors, the Keynumber substitution can be repeated. Each execution will produce a new set of Keynumbered underlinings, restarting with 111, 222,.... SODA will reprocess Keynumber input lines from previous correction attempts, and then match the most recent input lines with the most recent set of Keynumbered statements. The user is allowed a maximum of five key number correction attempts to the original SODA transaction. If transaction errors still occur, the user must scratch his input with

(SMI) (logical not) STOP (ENTER)

then start again.

(SMI) STARTOVER (logical not) F (ENTER)

and reenter his transaction.

If the user discovers that he has omitted any lines from the end of his original input, he should enter those lines

first, and then enter any Keynumber input in a separate physical transmission. If the discovery is made after some Keynumber input has been entered, a standard replacement-by-line-number may be used. However, repetitive corrections are more confusing than starting over. The user is advised to scratch his multiple-corrected input with

(SMI) (logical NOT)STOP (ENTER)

and then reenter.

When an error condition is detected by the logic statement during execution, the POOL operation ERR $field,'message' should be executed. This will not terminate execution, but it will display the message, save column numbers for subsequent underlining, and continue. Whenever execution is terminated, SODA will not consider this a successful update. It will, instead, display the update transaction underlining each field referred to in an ERR statement, as described above.

If the user omits some characters, or types some extra ones, the Keynumber substitution is inadequate, since it does not change the size of the transaction. For example, the user's input might be:

(ERASE) (SMI) FILE TEST360 REPORT REMOTE (ENTER) (NEWLINE)

(SMI) AN00147XY2234ABC00K97 (logical NOT)F (ENTER)

A B

Two errors have been made: one character has been omitted at A, and a nonnumeric character has been entered at B. The logic statement will be designed to detect both of these, as shown above, with the following output:

111 NOT NUMERIC

222 SUBSET NOT FOUND

---TEST360--REMOTE---

```
AN00147XY2234ABC00K97
      222       11111
```

Keynumber substitution cannot correct the offset caused by the omitted character. SODA will accept the following entry, which does expand the transaction:

(SMI) INSERT = Z AT 2234, 111=00597 (logical NOT) F (ENTER)

SODA processes INSERT by matching the pattern 2234 with the text, left to right, first with columns 1-4, then 2-5, 3-6.... A match occurs with 10-13, so the text starting at column 10 is shifted right to accommodate the insertion "Z". The text then reads:

AN00147XYZ2234ABC00K97

and 111=00597 is processed as usual. The insertion and pattern are both considered value words, and can be of any length, and can be literals. In particular, the pattern should be at least long enough to be unique in a left-to-right scan to the point of insertion. SODA also accepts

```
CHANGE
      = string AT pattern
DELETE
```

It is possible to achieve the same effect as 111=00597 by saying CHANGE = 5 AT K97. If the user had entered

```
AN00147XYZZ22234ABC00K97
          to be removed
```

he could correct it by entering

(SMI) DELETE = XX at Z22 111=005 (logical NOT) F (ENTER).

The pattern is used to determine the point of deletion, and the string (XX) is used to specify the number of characters to be deleted. Any characters may be used in the string, even the string to be deleted (Z2).

4.3.2 Standard Line Replacement Capability

The TP Monitor retains the user's input on a disk input queue. Each of the original transmissions is retained as a line, padded with blanks to the defined line size of the console. The lines are numbered Ø1, Ø2, ... Each numbered line can be replaced on the input queue, by use of the format described in section 2.3.3.3. That capability can be used to correct an update transaction. If the input were

(SMI) FILE TEST360 REPORT REMOTE (ENTER) (NEWLINE)

(SMI) ANØØ147XY2234ABCØØK96 (logical NOT) F (ENTER)

A B

with the errors detected as above. The user should retype the complete line Ø2 of his input in its correct form:

(SMI) (logical NOT) Ø2 ANØØ147XBA2234ABCØØ597 (ENTER)

The TP Monitor will substitute this for line Ø2 of the original input. The request for SODA may now be entered again. (There cannot be two (logical NOT) symbols in the same transmission.)

(SMI) (logical NOT) F (ENTER)

SODA will receive the corrected input, will not find any Keynumber substitutions, will discard its record of the underlined fields, and will process the transaction. A complete picture of the display screen is given as a summary.

(In) ⌀FILE TEST360 REPORT REMOTE⌀ (NEWLINE)

(In) ⌀ANØØ147XY2234ABCØØK97 (logical NOT) F⌀

(Out) 111 NOT NUMERIC

(Out) 222 SUBSET NOT FOUND

---TEST360--REMOTE---

(Out) AN$$147XY2234ABC$$K97

(Out) 222 11111

(In) $(logical NOT)$2AN$$147XBA2234ABC$$597

(In) (logical NOT)F$

(Out) INPUT SCRATCHED. READY FOR MORE.

If the user desires, he may attempt to save himself some typing by manipulating the cursor to take advantage of any existing characters on the screen. This is often more tedious than retyping, and it is not a recommended technique.

4.4. Output Messages and Actions

In contrast to a terminal inquiry, the goal of SODA is an updated file record, instead of readable text. The logic statement may log or print any text the analyst finds desirable. For example, he may wish to show the user the before and after contents of each field he updates. If the POOL statement LOG or PRT is used, the line is displayed immediately without use of the output queue (all output described thus far in section 4 is of the immediate type). If the POOL statement PT2 is used, the line will be placed on the output queue. The user will then be required, at the end of the update, to use conversational display commands (see section 2.3) to display his output, and/or scratch it before entering another update.

SODA will display at least one status message on every execution. At normal completion:

INPUT SCRATCHED. READY FOR MORE.

If the user enters the UPDATE operator:

ALL UPDATED RECORDS WRITTEN ON filename.

is displayed to confirm the transfer of records to the master file. The response to an UPDATE of specific keys is RECORD record-key WRITTEN ON filename.

If the user enters the CANCEL operator:

ALL UPDATES TO filename CANCELED
SIGNOFF ACCEPTED filename.

is displayed to confirm the deletion of previous updates. The response to cancellation of specific keys is:

CHANGES TO RECORD record-key CANCELED.

A self-explanatory diagnostic is followed by:

INPUT RETAINED FOR CORRECTION

TYPE YOUR CORRECTIONS THEN TYPE ¬F, ENTER.

IF YOU WANT TO BEGIN AGAIN WITHOUT CORRECTING TYPE ¬S

ENTER - -

TYPE STARTOVER ¬F ENTER - -

THEN TRY AGAIN

LOGIC STATEMENT statement name NOT FOUND.

4.5 Restrictions

Because of the nature of an on-line program, several restrictions are imposed by SODA.

4.5.1 File Access

SODA handles only updates to single files resident on disk in indexed sequential format. This is really a restriction on the design of a logic statement and update transaction. If the user signs off on a file at any point in time, he may update a new file. Similarly, two users at separate terminals may each be updating the same or different files independently.

4.5.2 Transaction Input

Only one update transaction may be entered at a time. Batching is not permitted in that the first transaction must be completely processed before a second is entered. Several terminals may be using SODA according to the standard TP rules (see introduction).

4.5.3 Logic Statements

SODA will not compile logic statements. A logic statement must exist in the library at the time of the request. The following types of logic statement actions are illegal in SODA: range updating, indirect addressing.

4.5.4 Error Reporting

SODA must be informed of errors detected by a logic statement. The design of the statement must use the POOL operation

```
ERR   $field,'message'
```

upon detection of an error. See Volume III, File Maintenance (FM) for coding details. If this operation is not performed at any time during the execution of a logic statement, SODA will assume a successful update occurred, place the record on the hold file, scratch the input and will not provide any correction aids. If, in fact, the logic statement did log an error without use of ERR, and omit part of the transaction processing, the user will be required to cancel the update and reenter his input.

4.6 Auxiliary Operations While Using SODA

When building update records in a hold file using SODA, it is not necessary to release SODA by a CANCEL or UPDATE command before requesting another program service (such as QUIP) from the Supervisor. After processing each update record, SODA returns to an idle (wait) status and its core requirement reduces to a minimum nucleus size (see Installation Manual for core sizes). At this point in time a user could, for example, run a QUIP query and then return to his SODA update operations. When SODA is requested again

(¬ F), the Supervisor reactivates it, retaining the hold file being built. A hold file is maintained on a file basis. This means that if many terminals are updating the same file (data base), a common hold file is used by all of them. An UPDATE request by any terminal user, in effect, will update the file for all users.

4.7 Step-by-Step Example of SODA Execution

Following is a step-by-step procedure for using SODA to add to a field in the fixed set, and to replace a field of a specified subset of a periodic set. The update transaction will require 25 characters as follows:

1	logic statement type
6	record Key
4	numeric field to be added
1	alphabetic field to be replaced
13	identifier for subset

Place on the FM logic statement library the following (note this operation may <u>not</u> be performed via SODA):

```
$FMS/LIB,TEST360

$AR,SAMPLE,1

$ASP,SAMPLE,A,8Ø

$KEY,2,7,C1

$NRAD,8,11

$CODE,12,12

$SUB,13,25

        POOL

        BNR BADKY
```

```
        MNU $NRAD,W1/4
        COA $NRAD,W1/4
        BEQ AD
        ERR $NRAD,'NOT NUMERIC'
        BRA NOAD
AD      ADD W1/4,PERS,PERS
NOAD    POV $SUB,MECLQ,BADSS
        MAL $CODE,MEDEP
        HLT
BADSS   ERR $SUB,'SUBSET NOT FOUND'
        HLT
BADKY   ERR $KEY,'RECORD SHOULD EXIST ON FILE'
        DDR
        END
```

From an inactive console enter:

(ERASE) (SMI) (logical NOT)STOP (ENTER)

This clears the input queue.

Enter:

(SMI) FILE TEST360 REPORT SAMPLE (NEWLINE)

AN00147Ø875D1Ø5M (logical NOT)F (ENTER)

This 54-character transmission identifies the file, identifies the report, enters an update transaction and requests execution of SODA. The TP Monitor will respond:

EOM RECEIVED and subsequently FMSODA STARTED.

At this point processing begins, and it will complete as a successful update to the hold file with the response:

INPUT SCRATCHED. READY FOR MORE.

This completes the processing of the first transaction. A second transaction to the same file may now be entered.

(ERASE) (SMI) AN525910K63D105M (logical NOT)F (ENTER)

This 27-character transmission enters an update transaction, and a request for execution. It does not identify a file or report, so these will be the same as in the previous update: TESTFIL and SAMPLE. The TP Monitor will respond as above, but this time a nonnumeric character has been entered. The logic statement will detect this, execute the ERR operation instead of the ADD:

111 NOT NUMERIC

The logic statement will continue, and upon completion, SODA will add to the display:

AN525910K63D105M
1111

The record is <u>not</u> placed on the hold file, even though the subset '105M' was updated with the code 'D'. The user may correct the bad field as follows:

(SMI) 111=0563 (logical NOT)F (ENTER)

The TP Monitor adds this to the original input and executes SODA with the standard EOM and STAPT responses. Internally, SODA receives 160 characters:

AN525910K63D105M (55 blanks)

111=0563 (72 blanks)

SODA retains data on the transaction size, and thus where corrections begin, and also retains data on columns which

are referred to by the Keynumber 111. The characters Ø563 replace ØK63 and the logic statement is executed again. This time the update is successful with the response:

INPUT SCRATCHED. READY FOR MORE.

This completes the second update. More could be entered, but for this example, we are ready to sign off:

(ERASE)(SMI) UPDATE ALL (logical NOT)F (ENTER)

The TP Monitor will again respond with EOM and received FMSODA STARTED. The hold file records for TEST36Ø will be located and written back to the master file, with the following response:

ALL UPDATED RECORDS WRITTEN ON TEST36Ø

SIGN OFF ACCEPTED TEST36Ø.

The console is now free for any use.

SECTION 5

EDIT

This section describes the function, capabilities, and user language specifications of the EDIT component. The primary function of EDIT is the on-line creation, modification and management of NIPS source programs. These programs can be error-scanned by the actual NIPS component editing routines, corrected, and submitted for batch execution from the terminal. The user can write or modify a NIPS program, have its language and format verified by the appropriate NIPS component, (FS, FM, RASP, OP, or QUIP) make any recommended corrections, and submit it for batch execution, all on-line.

EDIT also provides for on-line management and batch submission of data other than NIPS source programs, if the data is stored in 80-character card images. The user may also request verification of ALC source code.

5.1 General Operation

The EDIT component processes the user's source material through the use of data sets and message queues. The terminal user gives EDIT input data and commands by entering them from the terminal to the Input Message Queue (IMQ). EDIT is initiated when the user enters an ¬E command. Through use of the EDIT commands (section 5.4) that were entered, the user directs the actions of EDIT. If any commands are not intelligible to EDIT or are otherwise in error, they will be listed on the Edit Message Queue (EMQ) followed by error messages. EDIT will take no actions if errors are found in any command. If no errors are found, EDIT will execute the commands entered by the user. Source material entered from the terminal or requested from libraries or sequential data sets will be placed on a temporary data set. Modifications will be made as requested and an audit trail of commands executed will be written on the EMQ. If a NIPS component has been requested to perform a language check of the source material, the specified component will be called. Any messages produced by the

component will also be placed on the EMQ. If the user has required ALC source code verification, the assembler is envoked and any messages produced will be placed in the EMQ.

The terminal user can also request EDIT to list information from libraries or the temporary data set being maintained. This information will be written to the Output Message Queue (OMQ).

When EDIT has completed the requested actions, the terminal user can page through the contents of both the EMQ and the OMQ.

5.2 General Concepts

The following paragraphs describe various concepts that apply to EDIT component processing.

5.2.1 Terms Used in EDIT

Certain terms used throughout this section have special meaning to EDIT processing.

A terminal session consists of everything done from the time a user initially sits down at a terminal and starts using the EDIT component until he terminates processing and leaves the terminal. A sample terminal session appears in section 5.8, Terminal Session Example.

An entry to EDIT consists of all statements entered at the terminal for one execution of the EDIT component. An entry begins with the first line entered to the IMQ and ends with the ¬E command to execute EDIT.

A WORK file is the name given to the temporary data set that contains the source material being modified or otherwise manipulated. All source data entered at the terminal or requested from other sources will reside on the WORK file until a disposition is specified.

A pass of the WORK file consists of copying the user's material onto a second temporary data set, resulting in a new WORK file. Normally, all actions requested involving

the WORK file will be performed in a single pass of the WORK file.

5.2.2 SEQ/NOSEQ Mode

The WORK file exists in one of two modes, sequenced or nonsequenced (SEQ or NOSEQ, respectively), and consists of 80-byte records. If the WORK file is in SEQ mode, bytes 73-80 of each record contain a sequence or line number; and references to the WORK file are based upon the line number physically contained in each record. If the WORK file is in NOSEQ mode, references to the WORK file are based upon a record count; and all 80 bytes of each record are assumed to contain data. In NOSEQ mode, the first WORK file record is referenced by line number 1, the second record by line number 2, and so on.

5.2.3 Modification of Retrieval Data

Whenever there is an existing WORK file, commands referencing line numbers are assumed to apply to the existing WORK file. When there is no existing WORK file (such as the initial entry to EDIT in a terminal session), commands referencing line numbers are valid only if they are preceded by a retrieval command that accesses a library or SAM data set. In this case, the lines affected by these commands are the lines in the material being retrieved. If there is an existing WORK file to modify and a retrieval command is issued, the material being retrieved may not be modified in this pass at the WORK file. Commands referencing line numbers apply to the WORK file lines. The material being retrieved is simply inserted unchanged into the appropriate spot in the WORK file where it may be modified in a later pass.

5.2.4 Permanent Data Sets

The user may access libraries (partitioned data sets) or disk resident SAM (sequential) data sets. To be referenced by EDIT, these data sets must be cataloged; and must either contain fixed, blocked, 80-byte records or contain undefined records, each containing 800 bytes formatted as ten 80-byte

records. In addition, only one library may be accessed in any one entry to EDIT, though it may be accessed for both input and output. The disk packs containing the libraries or other files referenced by the EDIT user must be mounted prior to entry into EDIT.

5.2.5 Temporary Data Sets

In certain cases, the user may find it necessary to temporarily save portions or all of the WORK file. To do this, he may issue commands which cause EDIT to create additional temporary data sets. Each temporary data set must be given a unique name by the user and must be deleted by the end of the terminal session in which it was created. A maximum of five temporary data sets per terminal may exist at any one time. If the user has created five temporary data sets and wishes to create another, one of the existing temporary data sets must be deleted before the new one may be created.

5.3 Language Description

The EDIT language consists of commands with associated free format, keyword parameters. When a command is entered, the first position in the input record must contain a slash (/) followed immediately (no intervening blanks) by the command. If the slash(/) is followed by at least five blanks, the line will be considered a null line on the IMQ and skipped. This can be used to delete an undesired EDIT command from the IMQ without creating a blank line which might cause an error. The associated parameters must be placed on the same input record as the command. At least one blank or comma must be used to separate parameters. Additional blanks or commas between parameters are optional. Blanks or commas may not precede or follow special characters (=, -, or +) within parameters nor may they be embedded anywhere within the parameters. Except where otherwise indicated, the parameters may be entered in any order. Each command must be entered as a separate record.

Almost any combination of commands may be entered in one entry to EDIT. The few invalid combinations of commands are indicated under the appropriate command paragraphs.

Any command that modifies lines in the WORK file or adds lines to the WORK file contains a line number parameter. Each line in the WORK file is associated with a line or sequence number. The line number parameter in a command designates the location in the WORK file where the modification or addition is to occur. Commands referencing line numbers must be entered in ascending line number sequence since all modifications or additions are made in a single sequential copy of the WORK file.

5.4 Commands

There are 17 commands in the EDIT language:

ALLOCATE	GET	MOVE	SIGNOFF
CHANGE	HOLD	PUT	SIGNON
DELETE	INSERT	RESEQ	SUBMIT
FORMAT	LIST	SETUP	UTIL
			VERIFY

These commands may be grouped into four categories: control, input, modification, and output. The control commands are ALLOCATE, FORMAT, SIGNON, SIGNOFF, SUBMIT, UTIL, and VERIFY.

The input commands are GET, MOVE (to the WORK file), SETUP, and SIGNON. Modification commands consist of CHANGE, DELETE (of WORK file lines), INSERT, and RESEQ. The other commands, DELETE (of anything other than WORK file lines), HOLD, LIST, MOVE (from the WORK file), and PUT, are output commands. Note that the SIGNON, the DELETE and the MOVE commands have been listed under two different categories. SIGNON is used to initiate a terminal session, it is a control command; when it is immediately followed by user supplied terminal input to create a WORK file, it is an input command. The category to which the DELETE and MOVE commands are assigned depends on the options specified in the command.

In one pass at the WORK file, output commands other than an initial HOLD command (section 5.4.8.2, HOLD Command) are

executed after modification commands and, consequently, should be entered after the modification commands which affect the lines being outputted.

5.4.1 Library Name Default Option

The library name parameter on library-referencing commands such as GET or PUT will default to the last library referenced by an EDIT command in this terminal session. Consequently, the library name parameter in a command must be specified only if the command is the first library referencing command in the terminal session or if a library other than the last one referenced is desired. However, the library name parameter may be specified whenever valid.

5.4.2 Line Number Indexing

In certain instances a SEQ mode WORK file may have multiple lines all with the same line number (see section 5.4.7.3, INSERT Command). When this occurs, the user may reference the second or later occurrences of the line number by specifying the line number plus an index factor (nnnn+ii). The first occurrence of the line number is specified by the line number only (nnnn) while the second occurrence is specified by an index of one (nnnn+1), and so on. If indexing is specified and there are no multiple occurrences of the line number, the command will be rejected. Also, indexing is invalid in NOSEQ mode.

Note: Whenever a line number is specified in the following paragraphs and indexing is valid, line number plus index is acceptable even though not specifically stated.

5.4.3 Zero Line Numbers

Zero is a valid line number. For a command that places data in the WORK file, such as a GET or INSERT, a zero line number causes the data to be placed in front of the data in the WORK file. Data that is placed in a SEQ mode WORK file, through an INSERT at Ø command, will contain line number Ø. A CHANGE or line DELETE command may modify the first line of this inserted data by referencing line number Ø and other

lines by referencing line number Ø+ii. A GET, INSERT, or MOVE to line number Ø would place the data in front of the zero lines, while these commands to line number Ø+1 would place the data following the first line with line number Ø.

5.4.4 Command Description Conventions

In the command description sections below, several conventions will be followed to describe the commands in the EDIT language. This paragraph discusses these conventions. Only that portion of the command word which is underlined must be entered though the entire command word may be entered. Parameters enclosed in square brackets [ ] are optional and may be omitted. If one of the options within the square brackets is entirely underlined, it is the default option and will be assumed if the parameter is omitted. Capitalized options are keyword and must be entered as shown; lowercase options are replaced by the appropriate value. Where a keyword parameter is partially underlined (MEMBER=name), only the underlined portion of the keyword must be entered though the entire keyword may be entered (either MEM=name or MEMBER=name is valid). Where a vertical list of options is specified, only one option may be chosen. If the entire parameter is not enclosed in square brackets [ ], the parameter must be specified.

Note: There will not be an underline under the slash which precedes commands but only under the characters of the command that must be entered.

5.4.5 Control Commands

The control commands are ALLOCATE, FORMAT, SIGNON, SIGNOFF, SUBMIT, UTIL, and VERIFY. These commands perform various controlling functions not covered under the other command categories.

5.4.5.1 ALLOCATE Command

The ALLOCATE command is used to create a temporary or permanent data set. The user specifies the name and size of the data set.

5.4.5.1.1 ALLOCATE Command Parameters

```
          tempname [SIZE=nnnn]
/ALLOCATE
          permname VOL=SER=volid [UNIT=type]

          [RECFM=cc] [LRECL=nnnn] [BLKSIZE=nnnn]

           CYL=nnn
           TRK=nnnn  [EXT=nnn][DIR=nn
           BLK=nnnnnn
```

/ALLOCATE — The ALLOCATE command is entered whenever the user wishes to allocate a SAM temporary file to contain all or part of the WORK file, or a permanent SAM data set, or a permanent partitioned data set (library).

tempname — The tempname parameter must be entered to specify a unique name for the temporary data set. This temporary name must follow standard System/360 OS conventions; i.e., be eight or less characters in length, start with an alpha character, contain no special characters or embedded blanks, and it may not be one of the EDIT component reserved words (see section 5.9, EDIT Component Reserved Words).

SIZE=nnnn — The user may specify a size for the temporary data set. If a size is not specified, the size of the current WORK file will be used. The number specified in the size parameter is the maximum number of 80-byte records that the user expects to place in the temporary data set.

Unless the entire WORK file is to be moved into the allocated temporary data set, it is best to specify the SIZE parameter. If the SIZE parameter is not

	specified, a larger temporary data set than required will be allocated.
permname	The permname parameter must be entered to specify a unique name for the data set. This name must follow standard System/360 OS conventions. The length is limited to 44 characters. There are no restrictions concerning EDIT reserved words.
VOL=SER=volid	The direct access volume on which the data set is to be allocated. The volume must be mounted.
UNIT=type	The device type of the volume specified. If omitted, it will default to 2314.
RECFM=cc	The desired record format of the data set. The legal values are: F (fixed), FB (fixed blocked), V (variable), VB (variable blocked), and U (undefined). The default is FB.
LRECL=nnnn	The logical record length of each record in the data set. It defaults to 80.
BLKSIZE=nnnn	The block size of each block in the data set. It defaults to 800.
CYL=nnn TRK=nnnn BLK=nnnnn	The space parameters indicate how many cylinders, tracks, or user-specified blocks to allocate. The default is three cylinders.
EXT=nnn	The number of cylinders, tracks, or user-specified blocks to be allocated for each extent. The default is 0.
DIR=nn	The number of directory blocks to be associated with the PDS. This parameter must be present when allocating a partitioned data set. If absent, a SAM file is allocated.

5.4.5.1.2 ALLOCATE Command Restrictions and Considerations

A maximum of five temporary data sets may exist per terminal at any one time. If there are five temporary data sets, a DELETE command to delete one of them must be issued before an ALLOCATE command may be issued.

If it is necessary to extend the ALLOCATE command onto more than one line of the IMQ, end the line with a comma and begin the next with /A. For example:

```
/A NIPSTEST VOL=SER=DA0010 UNIT=2314,
/A CYL=5 DIR=2
```

5.4.5.2 FORMAT Command

The FORMAT command is used to specify automatic formatting of data entered by the user at the terminal.

5.4.5.2.1 FORMAT Command Parameters

```
/FORMAT  [OFF   ]
         [POOL  ]
         [n1,n2,...ni]
```

/FORMAT — The FORMAT command is used to specify automatic formatting of data entered by the user at the terminal. The FORMAT command defines the automatic columnar spacing desired by the user.

OFF — (Default value). Specifies that any lines entered by the user following a CHANGE or INSERT command are to be placed in the WORK file exactly as entered at the terminal (other than the inserting of line numbers when in SEQ mode).

POOL — Specifies that the terminal input lines entered by the user are to be automatically reformatted using POOL language conventions. That is, a non-

blank word starting in column 1 is to be shifted to column 6, the first nonblank word following a blank or blanks is to be shifted to column 16, and the next nonblank word following a blank or blanks is to be shifted to column 21. For example, the following two lines of terminal input would be entered into the WORK file in the format indicated below:

Terminal Input Lines:

```
Column:    1
           IDPNT MAL        RECID,W1/10
              PRT W1/10
```

Reformatted WORK File Lines:

```
Columns:   1   6      16    21
               IDPNT  MAL   RECID,W1/10
                      PRT   W1/10
```

n1,n2,...,ni

Similar to the POOL option, except that the user specifies the columnar spacing. The blanks in the terminal input lines act as delimiters to the words causing them to be aligned in the columns specified by the user. The numbers n1 through ni must be entered in ascending sequence and are used as the starting columns instead of the POOL language standard of columns 6, 16, and 21. For example, if the user specified 10,15,25,47 as his option, a nonblank word starting in column 1 of the following terminal input would be aligned so that it began in column 10, the first nonblank word following a blank or blanks would be shifted to column 15, the second nonblank word following a blank or blanks would be shifted to column 25, and the third word following a blank or blanks would be aligned in column 47.

5.4.5.2.2 FORMAT Command Restrictions and Considerations

Once a format has been specified on a FORMAT, SETUP, or SIGNON command, it remains in effect for all terminal input until another FORMAT or SETUP command is entered. The FORMAT command may be entered at any time other than following a last command such as VERIFY or SUBMIT.

If there are more words on a terminal input line than there are specified columns, the spacing between the words following the last word for which a column is specified remains the same as it was entered on the terminal input line. For example, comments may be entered on instructions when the POOL format is in effect and the spacing will be as indicated in the example below:

Terminal Input Lines:

```
Columns:  1
           BNR NEW    NEW RECORD CHECK
           MAL 'DUPE RECORD',W1/15
           LNK  ERRPNT  LINK TO ERROR ROUTINE
          NEW MAL   $A1,A1
```

Reformatted WORK File Lines:

```
Columns:
     1    6      16      21
                 BNR     NEW   NEW RECORD CHECK
                 MAL     'DUPE RECORD',W1/15
                 LNK     ERRPNT  LINK TO ERROR ROUTINE
          NEW    MAL     $A1,A1
```

5.4.5.3 Control SIGNON Command

The SIGNON command must be the first command entered at the start of a terminal session. It ensures that the last user of the EDIT component at this terminal did SIGNOFF and that there are no existing temporary data sets and no existing WORK files. If the previous user of this terminal did not issue a SIGNOFF command, the interpret scan of the current user's commands will be terminated immediately and a diagnostic message will be sent to the terminal so that corrective action may be taken. The corrective action may

consist of contacting the previous user for instructions or simply issuing a SIGNOFF command for the previous user.

For an explanation of the SIGNON command parameters, see section 5.4.6.4, Input SIGNON Command.

5.4.5.4 SIGNOFF Command

The SIGNOFF command terminates the terminal session and must be entered as the last command of a terminal session. The SIGNOFF command may delete all existing temporary data sets.

5.4.5.4.1 SIGNOFF Command Parameters

/SIGNOFF [DELETE]

/SIGNOFF	Terminates the terminal session and must be entered as the last command of a terminal session.
DELETE	Specifies that any existing temporary data sets are to be deleted during SIGNOFF processing. It is an error condition if the DELETE parameter is not specified and there are any temporary data sets. It is not an error condition if the DELETE parameter is specified and there are no temporary data sets to be deleted.

5.4.5.4.2 SIGNOFF Command Restrictions and Considerations

The SIGNOFF command must be the last command entered in a terminal session. All temporary data sets must be deleted prior to or during SIGNOFF processing.

5.4.5.5 SUBMIT Command

The SUBMIT command specifies that an entire batch job has been formatted on the WORK file and the job is to be submitted for batch processing. The SUBMIT command deletes the WORK file.

5.4.5.5.1 SUBMIT Command Parameters

/SUBMIT [SAVEWORK]

/SUBMIT	Specifies that an entire batch job has been formatted on the WORK file and the job is to be submitted for batch processing. The SUBMIT command deletes the WORK file.
SAVEWORK	Specifies that the WORK file is to be saved and not deleted as per the normal action.

5.4.5.5.2 SUBMIT Command Restrictions and Considerations

The entire job stream including the JOB card and appropriate JCL cards must be contained in the WORK file.

The WORK file is deleted by the SUBMIT command.

The SIGNOFF command is the only command that may follow a SUBMIT command in one entry to EDIT.

5.4.5.6 UTIL Command

The UTIL command requests that a specific OS-type utility function be performed. Successful completion and diagnostic messages are placed on the Edit Message Queue.

5.4.5.6.1 UTIL Command Parameters

```
        LOCATE   DSNAME=name
        CATLG    DSNAME=name    VOLUME=unit=volid
/UTIL   UNCATLG  DSNAME=name
        SCRATCH  DSNAME=name    [VOLUME=unit=volid] [PURGE]
        RENAME   DSNAME=name [NEWNAME=name  [VOLUME=unit=volid]
```

/UTIL	The UTIL command specifies that a utility function has been requested.
LOCATE	Returns the volid to which the data set is cataloged
CATLG	Catalogs the data set
UNCATLG	Uncatalogs the data set
SCRATCH	Scratches the data set
RENAME	Renames the data set
DSNAME=name	The data set upon which the utility function is to be performed
VOLUME=unit=volid	The volume identification of the volume containing the data set. Unit is the device type, e.g., 2314, 2311, 2400. Volid can be a single volume ID or can be a list in the form of (volid1,volid2,volid3), where the maximum number of volumes is 5.
NEWNAME=name	The new name of an old data set. It is a parameter on the RENAME function only. The name must follow standard System/360 OS conventions.
PURGE	Permits scratching of a data set even if the retention date has not expired. It is an option on the SCRATCH function only.

5.4.5.6.2 UTIL Command Restrictions and Considerations

When using the CATLG function, the VOL parameter must be specified.

When using the SCRATCH and RENAME functions, the VOL parameter need not be specified if the data set is cataloged. The volumes containing the data sets must be mounted. No maintenance is done on the catalog in conjunction with these functions; i.e., all data set names cataloged are still cataloged after scratching or renaming.

If it is necessary to continue the UTIL entry onto another line of the IMQ, the first line must end with a comma, and the next line begin with a /U. For example:

```
/U SCRATCH DSNAME=NIPSM.JOBLIB,
/U VOL=2314=(VOLID1,VOLID2)
```

5.4.5.7 VERIFY Command

The VERIFY command causes the current WORK file to be passed to the specified NIPS component diagnostic routine or the assembler for analysis. Diagnostic messages are placed on the Edit Message Queue for paging by the user.

5.4.5.7.1 VERIFY Command Parameters

```
              FS
              FM
/VERIFY       RASP    FILE=name [VOL=volid] [LIBRARY=name]
              OP
              QUIP
              ASM
```

/VERIFY — The VERIFY command must be used to specify the NIPS component or the assembler so that the correct diagnostic routine will be executed.

FS FM BASP OP QUIP ASM	NIPS component containing correct diagnostic routine to be used. Assembler to be used.
FILE=name	Name of the NIPS ISAM file. This parameter is required since most of the diagnostic routines must access the appropriate NIPS ISAM FFT for which this source language routine has been written. (Not required if FS or ASM is used)
VOL=volid	The volume identification (VOL=volid) of the direct access volume containing the NIPS FFT, if the NIPS ISAM FFT is not cataloged or an uncataloged version is to be used. (Not required if ASM is used).
LIBRARY=name	If the VERIFY operation requires library tables or subroutines, and the library has not been previously referenced or specified in this terminal session, the library name must be specified, unless the library is the file associated library (filename + 1).

5.4.5.7.2 VERIFY Command Restriction and Considerations

If the VERIFY command is entered, it must be the last command specified in an entry.

5.4.6 Input Commands

The input commands are GET, MOVE to the WORK file, SETUP, and SIGNON. The GET and MOVE to the WORK file commands take the records from a library member or SAM data set and place them in the WORK file. The SETUP and SIGNON commands may create a new WORK file from user supplied terminal input records. Processing of an input command is completed before processing of another input command is initiated.

5.4.6.1 GET Command

The GET command retrieves data from a SAM data set or member of a partitioned data set and places it on the WORK file.

5.4.6.1.1 GET Command Parameters

```
/GET  SAM=name
      MEMBER=name [LIBRARY=name] [READ  ] [0[+ii]   ] [SEQ  ] [FORM=name]
                                 [UPDATE] [nnnn[+ii]] [NOSEQ]
                                          [END     ]
```

/GET	The GET command is used to retrieve data from either a cataloged SAM data set or a member of a cataloged library.
SAM=name	Specified when the data is being retrieved from a SAM data set.
MEMBER=name	Specified when the data is being retrieved from a cataloged library member.
LIBRARY=name	Specified when the data is being retrieved from a library member and the library has not been referenced by a previous command.
READ	Specified when the data is being retrieved from a library member and the member is being retrieved in the READ mode. The READ mode (default option) specifies that the member is being used for input only and there will be no attempt to issue a PUT command to this member name.
UPDATE	Specified when the data is from a library member and the member is being retrieved in the UPDATE mode. The UPDATE mode specifies that, after some modifications, the user intends to PUT or UPDATE this member under the same name. The EDIT component enqueues the member name so

	that other users may not access it. The user must issue a GET in the UPDATE mode before he may issue a PUT in the OLD mode. (See Section 5.4.8.5, PUT Command.)
O[+ii] nnnn[+ii] END	Specifies the line (and index) number of existing WORK file, after which the retrieved data is to be replaced. END is the default option and indicates the retrieved data is to be placed at the end of the WORK file.
SEQ	Specifies sequenced mode (default option).
NOSEQ	Specifies nonsequenced mode. If there is no input WORK file, and the mode has not been specified, and a GET command is issued, the GET command parameter sets the mode. Succeeding GET commands must specify the same mode. If a 'GET' is not issued the mode defaults to SEQ.
FORM=name	Specifies the 1-8 character entry point name of the user written format subroutine. The subroutine must be a member of the same library as the member. If the data being retrieved is from a SAM data set the library must be specified by a previous command. (See section 5.11 for complete description of the User Written Format Subroutine).

5.4.6.1.2 GET Command Restrictions and Considerations

Processing of a GET command is completed before processing of another input command, such as an input MOVE command, is initiated.

If there is no input WORK file when a GET command is issued, the retrieved data is treated as an input WORK file and may be modified by the modification commands. If there is an input WORK file, data retrieved by a GET command is placed directly in the WORK file and modification commands

are processed against the input WORK file rather than the retrieved data.

Once a GET command in the UPDATE mode has been issued, no additional GET commands in the UPDATE mode may be issued until that member has been dequeued. The member is dequeued by a PUT command with the OLD option (section 5.4.8.5), a SETUP command with the FREE parameter (section 5.4.6.3), or a SIGNOFF command (section 5.4.5.4).

5.4.6.2 Input MOVE Command

The input MOVE command is used to move data from a temporary data set or the Output Message Queue to the WORK file. The MOVE command may also be used to move data from the WORK file to a temporary data set. For details on this use of the MOVE command, see the output commands subsection, section 5.4.8.4, Output MOVE Command.

5.4.6.2.1 Input MOVE Command Parameters

```
/MOVE  tempname  [nnnn[+ii]  -nnnn[+ii]]  TO WORK  [0[+ii]  ]
       OMQ       [          END         ]           [nnnn[+ii]]
                                                    [END      ]
```

/MOVE	Used to move data from a temporary data set or the Output Message Queue to the WORK file. The input MOVE command parameters must be specified in the order shown above.
tempname	Name of the Input Data set. The specified name may be that of a temporary SAM data set created by an ALLOCATE or HOLD command.
OMQ	The specified name of the input data set may also be OMQ for Output Message Queue.
nnnn[+ii]	Low line number of a selected portion of the input data set to be placed on the WORK file.

-nnnn[+ii]
-END

High line number of a selected portion of the input data set to be placed on the WORK file preceded by a dash, or END to indicate that all data to the end of the input data set is to be placed on the WORK file.

If the entire input data set is to be placed in the WORK file, this parameter is omitted and the keywords TO WORK immediately follow the input data set name.

If, however, only a selected portion of the input data set is to be placed on the WORK file, the range of line numbers encompassing the desired lines is specified following the input data set name and preceding the keywords TO WORK.

The specified range of line numbers may be either low line number, dash, high line number; or a low line number, dash, and END. In the first case, all the lines between and including the specified lines are placed in the WORK file, while in the second case, all the lines starting with the specified line and through the end of the input data set are placed upon the WORK file.

The interpretation of the specified range of line numbers is based upon the SEQ/NOSEQ mode of the input data set just as the interpretation of line numbers referring to the WORK file is based upon the mode of the WORK file. The mode of a temporary data set is the same as the mode of the WORK file at the time data was placed in the temporary data set.

The mode of the Output Message Queue (OMQ) is NOSEQ.

TO WORK	These two keywords are required to specify the type of MOVE commands. They are followed by line numbers specifying the location in the WORK file where the input data is to be placed. A line number specifies the data is to be placed immediately after the WORK file line.
0[+ii] nnnn[+ii] END	Specifies the line (and index) number of the existing WORK file, after which the retrieved data is to be replaced. END is the default option.

5.4.6.2.2 Input MOVE Command Restrictions and Considerations

The specified temporary data set or OMQ remains unchanged by the MOVE command.

If the temporary data set was created by an ALLOCATE command, an output MOVE command (section 5.4.8.4) placing data in the temporary data set must have been issued in a prior pass at the WORK file.

To use the MOVE OMQ TO WORK command, the user must have placed the Output Message Queue in hold (¬H) status. When this action is taken, EDIT cannot write on the OMQ. Therefore, during this EDIT pass, a regular LIST command cannot be issued. Only a LIST IMMED option is acceptable.

5.4.6.3 SETUP Command

The SETUP command reinitializes the EDIT component. Each entry to EDIT initializes a pass at the WORK file and each SETUP command initializes a pass at the WORK file. If a SETUP command is preceded by other commands in one entry to EDIT, processing of all commands preceding the SETUP command is completed before processing of the SETUP command is initiated. In this case, the SETUP command may initiate a second pass at the WORK file.

The SETUP command may be used to specify a different input WORK file, change the mode of the WORK file, change

the format of terminal input, or release an enqueued member name.

5.4.6.3.1 SETUP Command Parameters

```
/SETUP [USING=NEW     ] [SEQ  ] [FORMAT=OFF  ]
       [      tempname] [NOSEQ]          POOL  ] [FREE=name] [SIZE=nnnn]
       [      WORK    ]                 n,n,...,n]
       [      OLD     ]
```

/SETUP — The SETUP command is used to define the file to be used as the input WORK file.

USING= — Keyword which names the input WORK file specifically.

NEW — Specifies that there is no input WORK file since a new WORK file is being created from scratch. The new WORK file will be generated from user-supplied terminal input immediately following the SETUP command and/or data retrieved by later GET or MOVE commands. Terminal input following a SETUP command is valid only if the USING=NEW option is chosen (default option).

tempname — Specifies that the named temporary data set is to be used as the input WORK file. This, in effect, deletes the normal input WORK file and replaces it with the temporary data set. Since the temporary data set becomes the input WORK file, any commands referencing the WORK file lines in this pass actually reference the temporary data set lines.

WORK — Specifies that the normal input WORK file is to remain the input WORK file. The purpose of this option is to allow the user to change the SEQ/NOSEQ mode of a WORK file, to allow the user to dequeue a member name, or to allow the user to

make a second pass at the WORK file in one entry to EDIT.

This option is valid only if there is an existing WORK file.

OLD — Specifies that the old or backup file is to become the input WORK file, providing a "backup" capability. This essentially negates any action that was taken in the previous pass to modify the WORK file by the GET, MOVE, INSERT, CHANGE and DELETE commands. If this option is specified, there must have been a normal input WORK file at the start of the previous pass.

SEQ/NOSEQ — The SEQ/NOSEQ parameter sets the mode for the WORK file which then applies to any later commands referencing the WORK file.

If SEQ, the default option, is specified and there are terminal lines immediately following the SETUP command on the Input Message Queue, the EDIT component will place the line number 1 in the first line of the following terminal input. It will then increment the line number by 1 and place it in each succeeding line of terminal input until an EDIT language command other than FORMAT is encountered on the Input Message Queque.

Any data placed on the WORK file by a later GET, input MOVE, or other command is assumed to have valid sequence numbers.

FORMAT — The FORMAT parameter establishes the format for all following terminal input. For an explanation of the various options, see Section 5.4.5.2, FORMAT Command. Once a format is established, it remains in effect until a later SETUP or FORMAT command is encountered.

FREE=name	Specifies that the enqueued member name is to be dequeued. The specified name must have been enqueued through a previous GET command in the UPDATE mode (section 5.4.6.1). Once a FREE parameter has dequeued a member name, the user may not issue a PUT command with the OLD option until he has issued another GET command in the UPDATE mode.
SIZE=nnnn	Size of the input WORK file. The number specified is the maximum number of 80 byte records that the user expects. Default size is 500 80-byte records. This number is carried until another SETUP is entered.

5.4.6.3.2 SETUP Command Restrictions and Considerations

Only one SETUP command may be issued in a single entry to EDIT.

User-supplied terminal input may immediately follow the SETUP command on the Input Message Queue only if the USING=NEW option is chosen. Line numbers are automatically placed in these terminal input lines if the SEQ mode is specified. These terminal input lines are placed as the first lines in the WORK file being output from this pass.

5.4.6.4 Input SIGNON Command

The SIGNON command initializes a terminal session. It must be the first command entered and may only be entered as the first command in a terminal session. It establishes the mode of the WORK file and the format of terminal input.

5.4.6.4.1 SIGNON Command Parameters

```
/SIGNON  [SEQ   ][FORMAT=OFF      ] [SIZE=nnnn]
         [NOSEQ]       POOL
                       n,n,...,n
```

/SIGNON	Used to initialize a terminal session. The parameters on the SIGNON command have the same meaning and options as the equivalent parameters on the SETUP command (section 5.4.6.3.1).
SEQ/NOSEQ	The SEQ/NOSEQ parameter sets the mode for the WORK file which then applies to any later commands referencing the WORK file. If SEQ, the default option, is chosen, the EDIT component will place line numbers starting with 1 and incremented by 1 in the terminal input lines immediately following the SIGNON command.
FORMAT	The FORMAT parameter establishes the format for any terminal input following the SIGNON command. For an explanation of the various options of the FORMAT parameter, see Section 5.4.5.2, FORMAT Command. Once a FORMAT is established, it remains in effect until a later SETUP or FORMAT command is issued.
SIZE=nnnn	Size of the input WORK file. The number specified is the maximum number of 80-byte records that the user expects. Default size is 500 80-byte records. This number is carried until another SETUP is entered.

5.4.6.4.2 SIGNON Command Restrictions and Considerations

The SIGNON command may only be entered once in a terminal session and must be the first command entered. User-supplied terminal input may immediately follow the SIGNON command on the Input Message Queue. Line numbers are automatically placed in these terminal input lines if the SEQ mode option is chosen. These terminal input lines become the first lines in the WORK file being output on this pass.

If the user neglects to type the SIGNON command at the terminal, it is assumed it has been entered with all of the default values specified.

5.4.7 Modification Commands

The modification commands are CHANGE, DELETE of WORK file lines, INSERT, and RESEQ. These commands are used to change the contents of the WORK file.

5.4.7.1 CHANGE Command

The CHANGE command is used to modify or replace lines of data in the WORK file. If there is no input WORK file, the CHANGE command may be used to modify or replace lines of data being placed on the WORK file through a GET command.

5.4.7.1.1 CHANGE Command Parameters

/CHANGE nnnn[+ii] -nnnn[+ii] [△...△...△]
-END

/CHANGE	Used to modify or replace lines of data in the WORK file.
nnnn[+ii]	Must be the line number to be modified. Expressed by itself, specifies the modification of a single line.
-nnnn[+ii]	A dash, followed by a second line number, specifies the range of lines to be modified. The range of line numbers may consist of the lower and upper line numbers (nnnn-nnnn).
-END	A dash, followed by the keyword, END, specifies the lower line through the end of the WORK file (nnnn-END) is to be modified.

△..△...△

The CHANGE command may be used to modify or replace lines of the WORK file. The modification option (△...△...△) indicates that each occurrence of a given string of data (search argument) within the line or lines specified is to be replaced by a second given string of data (function). To use the modification option the user must first decide on a delimiter. This delimiter may be any character other than a blank or comma. Following the line numbers and at least one blank or comma, the user enters the delimiter, followed immediately by the search argument; repeats the delimiter to terminate the search argument and initiate the function; enters the function; and then again repeats the delimiter to terminate the function. For example, if the user wanted to modify line 10 of the WORK file by replacing any occurrence of MAL by MNU and decided to use a single quote (') as a delimiter, he would enter:

```
/C      10    'MAL'MNU'
```

The delimiter is not included as part of the search argument or function and may not be embedded within them. There must be three occurrences of the delimiter. The first occurrence defines the delimiter and indicates the start of the search argument. The second occurrence of the delimiter terminates the search argument and indicates the start of the function. The third occurrence of the delimiter terminates the function. Only one search argument/function pair may be specified per CHANGE command. Any blank or commas appearing between the delimiters will be treated as part of the search argument or function.

The search argument and function may be of different lengths. When they are, data on the WORK file line to the right of the characters being modified will be shifted to the right or left as necessary. Padding (with blanks) or truncation will occur only on the rightmost characters of the WORK file line. The rightmost character is in column 72 in SEQ mode and column 80 in NOSEQ mode.

If the CHANGE command is being used to replace entire lines of data on the WORK file, the only parameter entered on the CHANGE command is the line number or range of line numbers to be replaced. Then the new lines are entered at the terminal immediately following the CHANGE command line. The effect of the replacement CHANGE is that of a modification DELETE of the specified lines and an INSERT of the replacement lines at the higher specified line number. That is, the line or range of lines specified on the CHANGE command is deleted from the WORK file and the replacement lines are inserted at that point. If the WORK file is in SEQ mode, all the replacement lines are given the lowest line number without index specified on the CHANGE command. The number of lines being deleted does not have to match the number of lines being inserted. If a FORMAT is in effect, it will be applied to the replacement lines.

5.4.7.1.2 CHANGE Command Restrictions and Considerations

The modification and replacement options may not be mixed on one CHANGE command.

If the user desires to enter two or more search argument/function pairs for a single WORK file line, a

separate CHANGE command specifying the same line number must be entered for each pair. However, multiple CHANGE commands for the same range of lines are invalid. As the WORK file is processed sequentially, processing is completed to the higher line number of the first CHANGE command before processing of the lower line number of the second CHANGE command is initiated.

5.4.7.2 Modification DELETE Command

The modification DELETE command is used to delete lines from the WORK file. The DELETE command may also be used to delete temporary data sets or members from libraries. For details on these uses see, Section 5.4.8.1, Output DELETE Command.

5.4.7.2.1 Modification DELETE Command Parameters

/DELETE nnnn[+ii] -nnnn[+ii]
-END

/DELETE	Used to delete lines from the WORK file.
nnnn[+ii]	Must be line number to be deleted. Expressed by itself, specifies the deletion of a single line.
-nnnn[+ii]	A dash, followed by a second line number, specifies the range of lines to be deleted. The range of line numbers may consist of lower to upper line numbers (nnnn-nnnn).
-END	A dash, followed by the keyword, END, specifies the lower line through the end of the WORK file (nnnn-END) is to be modified.
	The lines are deleted from the input WORK file or, if there is no input WORK file, from the input retrieved by the last GET command issued in this pass.

5.4.7.2.2 Modification DELETE Command Restrictions and Considerations

The only restriction on the DELETE command is that when entering multiple DELETE commands, line number specifications must be in ascending sequence.

5.4.7.3 INSERT Command

The INSERT command causes data lines entered by the user at the terminal to be inserted at the indicated point in the WORK file.

5.4.7.3.1 INSERT Command Parameters

```
/INSERT   [ 0[+ii]   ]
          [ nnnn[+ii] ]
          [ END       ]
```

/INSERT	Used to insert data lines entered by the user at the terminal at the point indicated in the WORK file.
0[+ii] nnnn[+ii] END	Specifies the line (and index) number of the existing WORK file, after which the retrieved data is to be replaced. If the WORK file is in the SEQ mode, the new lines are given the line number (without the index factor) specified in the INSERT command. If END is specified, the new lines are given the same line number as the last line in the WORK file. The new lines to be inserted in the WORK file are entered from the terminal immediately following the INSERT command line.

5.4.7.3.2 INSERT Command Restrictions and Considerations

If an INSERT command specifying the same line number as a CHANGE or modification DELETE command is issued, the INSERT command must be entered following, not preceding, the CHANGE or modification DELETE command.

5.4.7.4 RESEQ Command

The RESEQ command resequences a SEQ mode WORK file.

5.4.7.4.1 RESEQ Command Parameters

/RSEQ INCR=xxxxxxxx

/RESEQ Used to resequence the user's work file.

INCR Specifies the increment to be used in the resequencing. The default is 1. The parameter value may contain a maximum of eight numeric digits.

5.4.7.4.2 RESEQ Command Restrictions and Considerations

The WORK file must be in SEQ mode for a RESEQ command to be valid. The first line of the output WORK file is line number 1, the second line is line number 2, and so on. The renumbering is done after all other modification command processing and prior to any output command processing. Consequently, any output obtained during a pass in which a RESEQ command has been issued shows the new line numbers.

5.4.8 Output Commands

The output commands are DELETE of other than WORK file lines, HOLD, LIST, MOVE from the WORK file, and PUT. With the output commands, the user may delete temporary data sets or library members, save data in temporary data sets, list data at the terminal, and create or update library members.

5.4.8.1 Output DELETE Command

The output DELETE command is used to delete temporary data sets or members of libraries. The DELETE command may also be used to delete lines from the WORK file (see section 5.4.7.2, Modification DELETE Command).

5.4.8.1.1 Output DELETE Command Parameters

```
/DELETE     tempname ... tempname
            ALL
            MEMBER=name [LIBRARY=name]
```

/DELETE	Used to delete temporary data sets or members of libraries.
tempname	Specifies the specific temporary file name to be deleted. The temporary file must have been created previously by an ALLOCATE or HOLD command. A maximum of five temporary names may be specified.
ALL	Specifies that all currently existing temporary data sets are to be deleted.
MEMBER=name	Specifies that a member of a library is to be deleted.
LIBRARY=name	Specifies the name of the library containing the member that is to be deleted, if no library has been referenced or if a different library is desired. The default option for the library sub-parameter is the last library referenced by a command in this terminal session.

5.4.8.1.2 Output DELETE Command Restrictions and Considerations

Temporary data sets must be deleted at the end of a terminal session either by this command or by the DELETE option on the SIGNOFF command.

5.4.8.2 HOLD Command

The HOLD command allocates a temporary SAM data set and saves the entire WORK file. The WORK file is deleted by the HOLD command.

5.4.8.2.1 HOLD Command Parameters

/HOLD tempname [RESEQ]

/HOLD	Used to allocate a temporary SAM data set and to save the entire WORK file in it.
tempname	A unique name that must be specified for the temporary data set. This name must follow standard System/360 OS conventions; i.e., is eight or less characters in length, starts with an alpha character, contains no special characters or embedded blanks, and may not be one of the EDIT component reserved words (see Section 5.9, EDIT Component Reserved Words).
RESEQ	Used to resequence the WORK file as it is being placed into the temporary file if the WORK file is in SEQ mode. In the resequenced file, the first record will be incremented by 1. If the WORK file is being resequenced, an actual copy of the WORK file will take place and the input WORK file will be deleted after the copy. If the file is not being resequenced, the WORK file will simply be renamed.

5.4.8.2.2 HOLD Command Restrictions and Considerations

The WORK file is deleted by a HOLD command. If a HOLD command is issued, it must be either the first or last command issued in one entry to EDIT. Two HOLD commands may be issued in one entry to EDIT as long as: one is the first command issued, some other commands create a new WORK file; and the second HOLD command is issued last.

The only commands that may immediately follow an initial HOLD command are a SETUP or a GET command.

A maximum of five temporary data sets may exist per terminal at any one time. If five temporary data sets exist, a DELETE command for one of them must be issued prior to a HOLD command.

5.4.8.3 LIST Command

The LIST command causes the specified data to be placed in the Output Message Queue (OMQ) or displayed directly at the terminal. The displayed data may be the directory or member of a cataloged library, the names of all temporary data sets, the contents of a temporary data set, the contents of the WORK file, or the volume table of contents (VTOC) of a direct access device.

5.4.8.3.1 LIST Command Parameters

```
                 [LIBRARY=name                ]
                 [           FS               ]
                 [           FM               ]
                 [COMPONENT= RASP             ]
                 [           OP               ]  [LIBRARY=name]
                 [           QUIP             ]  [FORM=name]
/LIST            [MEMBER=name                 ]                  [IMMED]
                 [SAM=name                    ]
                 [ALL                         ]
                 [tempname                    ]
                 [nnnn[+ii]  -nnnn[+ii]       ]
                 [           -END             ]
```

TERMINAL PROCESSING (TP)

[WORK
VOL=devicetype=volid]

/LIST — Used to place specified data in the Output Message Queue (OMQ) or to display data directly at the terminal.

LIBRARY=name — When used as an option, the names of all members listed in the directory of the specified library will be placed on the OMQ for paging by the user. Used as a subparameter of the COMPONENT option only if the desired library was not the one last specified by a command in this terminal session.

Used as a subparameter of the MEMBER option only if the desired library was not the one last specified by a command in this terminal session.

COMPONENT=FS, FM, RASP, OP, QUIP — Used to specify that only a selected portion of the member names listed in the directory of the specified library (the one last specified by a command in this terminal session or the one specified by the subparameter, LIBRARY=name) will be placed on the OMQ. The member names are those which contain source statements in the language of the specified NIPS Component (FS, FM, RASP, OP or QUIP). Non-NIPS source data may also be listed if an identifying class value was appended with the PUT command or by the UTSOURC utility program.

MEMBER=name — Used to cause the entire contents of the named member of the specified library (the one last specified by a command in this terminal session or the one specified by the subparameter, LIBRARY=name) to be placed on the OMQ.

SAM — Used to cause the contents of the named SAM data set to be placed on the OMQ.

ALL	Used to place the names of all currently existing temporary data sets created by the user in this terminal session on the OMQ.
tempname	Used to list the contents of the named temporary data set. The temporary data set must have previously been created by a HOLD or ALLOCATE command.
nnnn[+ii]	Used to place the line specified from the WORK file on the OMQ.
-nnnn[+ii]	High-order line number of a selected portion of the WORK file to be placed on the OMQ. Used in conjunction with the previous parameter (the low-order line number) to establish a range of line numbers to be listed.
-END	Used in conjunction with the low-order line numbers to place all lines in the WORK file from the line number specified to the end of the WORK file on the OMQ. If more than one LIST from the WORK file is specified in one entry to EDIT, the specified line numbers may not overlap.
WORK	Used to specify that the entire WORK file is to be placed on the OMQ (default option). Default option.
VOL	Used to request that the VTOC of the specified pack be placed on the OMQ. Devicetype is the type of direct access device, such as 2314. Volid is the volume serial number of the desired volume. LIBRARY is not a volid sub-parameter.
FORM=name	Used to specify the 1-8 character entry point name of the user written format subroutine. The subroutine must be a member of the same library as the member.

If the data being listed is from a SAM data set the library must be specified by a previous command.

IMMED — May be used with any of the above options and specifies that data generated by the LIST command is not to be placed on the OMQ but, instead, is to be sent directly to the terminal for immediate viewing by the user. If the LIST command with the IMMED option would generate more than one page of data, only one page is generated and a message indicating that only a portion of the data is being displayed is appended to the last line of the display. If the user then desires to see the rest of the data, he would have to reenter the LIST command without the IMMED parameter and page through the data after it was placed on the OMQ.

5.4.8.3.2 LIST Command Restrictions and Considerations

If the WORK file is being modified and listed in the same entry to EDIT, the line numbers on the LIST command refer to the input WORK file lines, but the lines placed on the OMQ reflect the corresponding lines as written on the output WORK file.

Only one LIST command with the IMMED option may be issued in an entry to EDIT.

5.4.8.4 Output MOVE Command

The output MOVE command is used to move data from the WORK file to a temporary data set created by the user through an ALLOCATE or HOLD command. The MOVE command may also be used to move data into the WORK file (see section 5.4.6.2, Input MOVE Command).

5.4.8.4.1 Output MOVE Command Parameters

/MOVE nnnn[+ii] -nnnn[+ii] TO tempname
-END

/MOVE	Used to move data from the WORK file to a temporary data set created by the user through an ALLOCATE or HOLD command.
nnnn[+ii]	Low-order line number of range of lines from WORK file to be moved. When used alone indicates one specific line to be moved.
-nnnn[+ii]	High-order line number of range of lines from WORK file to be moved.
-END	Used in conjunction with the low-order line number to specify that all lines from the line specified to the end of the WORK file are to be moved.
TO tempname	The keyword TO followed by the name of the temporary data set where the lines are to be placed. The temporary data set must have been created previously through an ALLOCATE or HOLD command and must be large enough to contain the specified lines.

5.4.8.4.2 Output MOVE Command Restrictions and Considerations

The WORK file is unchanged by an output MOVE command.

Any data already in the temporary data set is replaced by the output MOVE command.

If multiple output MOVE commands are specified in one pass at the WORK file, the specified range of line numbers to be moved may not overlap.

5.4.8.5 PUT Command

The PUT command stores the contents of the WORK file or a temporary data set on a library.

5.4.8.5.1 PUT Command Parameters

```
                                                                FS
                                                                FM
/PUT MEMBER=name [LIBRARY=name] [NEW] COMPONENT=RASP [RESEQ] [WORK    ]
                                [OLD]                OP      [tempname]
                                                     QUIP
                                                     cccc
```

/PUT	Used to store the contents of the WORK file or a temporary data set on a library.
MEMBER=name	Specifies the name under which this data is to be stored on the library. The name must meet standard System/360 OS PDS member name requirements.
LIBRARY=name	Specifies the library where the member is to be stored. It defaults to the last library referenced by a library referencing command.
NEW	Specifies that the member will be a new member. There may not already be a member with the name specified.
OLD	Specifies that the member will replace an existing member by the same name. There must be a member with that name to be replaced. In addition, the user must have issued a GET command in the UPDATE mode so that the member name is enqueued. A PUT command with OLD as an option dequeues the member name.
FS FM RASP	Specifies the NIPS component of the source language so that the information may be stored in the directory entry whenever the

COMPONENT=OP
QUIP
cccc

data being stored is NIPS source language material. The components that may be specified are FS, FM, RASP, OP, and QUIP.

If the material is something other than source statements for one of the above components, the user may specify a 1-to 4-byte value (cccc) to classify the type of data being stored. The use of the word NONE will set the component identification to blanks.

If OLD is specified, the user may omit the component parameter and it will default to whatever the old directory specified. If the user does specify the component when he has chosen the OLD option, the component specified by the user must be the same as the one currently in the old directory entry for the member.

RESEQ

Specified if the lines are to be renumbered starting with 1 and incrementing by 1 as they are placed on the library. The RESEQ option is valid only if the material is in the SEQ mode.

tempname

The name of the temporary data set of the WORK file from which the material being stored on the library is coming. If it is a temporary data set, the temporary data set name must be specified.

WORK

When the material being stored on the library is not coming from a temporary data set, WORK may be specified. This parameter then may be omitted since WORK is the default option.

5.4.8.5.2 PUT Command Restrictions and Considerations

The data set from which the material being stored is taken remains unchanged by the PUT command.

5.5 Summary of Commands

This command summary is provided in alphabetic sequence as a quick reference for users already familiar with the EDIT language. It contains a brief description and entry format for each command.

ALLOCATE

The ALLOCATE command initializes and names a temporary data set for temporary retention of WORK file data or allocates for future use a permanent data set. Temporary data sets cannot exceed five in number at any given time during a terminal session, and must be deleted at the termination of the session.

```
/ALLOCATE tempname [SIZE=nnnn]

/ALLOCATE permname VOL=SER=volid[ UNIT=type]
  [RECFM=cc][ LRECL=nnnn ][ BLKSIZE=nnnn]
       CYL=nnn    [EXT=nnn][DIR=nn]
       TRK=nnn
       BLK=nnnnn
```

CHANGE

The initial function of the CHANGE command is to replace one or more entire records of the existing WORK file.

The second function replaces only specified fields of a record. The delimiter character is taken as the first non-blank character following the CHANGE line number. By allowing a variable delimiter, the user is able to enter any string of characters between the unique delimiters. The field between the first and second occurrences of the unique character is the search argument, and the field between the second and third occurrences is the function. If the function is not the same size as the search argument, the record will be truncated or padded, as necessary, to 80 characters.

```
/CHANGE  nnnn[+ii] -nnnn[+ii] [△..△...△]
                   -END
```

DELETE (modification)

The DELETE (modification class) command causes those lines specified in the operand to be deleted from the WORK file.

```
                          -nnnn[+ii]
/DELETE    nnnn[+ii] -END
```

DELETE (output)

The DELETE (output class) command deletes a library member, a temporary data set, or all temporary data sets, as specified in the operand.

The DELETE command may also delete multiple temporary files within a single command.

```
           tempname ...  tempname
/DELETE    ALL
           MEMBER=name  [LIBRARY=name]
```

FORMAT

The function of the FORMAT command is to initialize automatic formatting of terminal input according to operand specifications, to change existing format specifications to those indicated in the operand, or to simply nullify existing format specifications.

```
/FORMAT  [OFF          ]
         [POOL         ]
         [n1,n2,...,ni ]
```

GET

The GET command retrieves data from a PDS member or a SAM data set. If a WORK file exists, the retrieved data is merged to a specific location therein; if not, the retrieved data constitutes a new WORK file. The retrieved data may be reformatted by a user written format subroutine. PDS

members are defined as update or read-only; SAM files are automatically read-only.

```
        SAM=name                                [READ  ] [nnnn[+ii]] [SEQ  ]
∠GET    MEMBER=name [LIBRARY=name]              [UPDATE] [END      ] [NOSEQ] [FORM=name]
```

HOLD

The HOLD command saves the current WORK file as a temporary data set with the name specified in the operand. The temporary data set need not have been previously allocated, and resequencing is optional.

```
∠HOLD   tempname [RESEQ]
```

INSERT

The INSERT command indicates that the following terminal input should be placed in the WORK file at the specified location.

```
∠INSERT  [nnnn[+ii]]
         [END      ]
```

LIST

The LIST command writes specified data either on the Output Message Queue for later display, or immediately to the terminal if the IMMED option is coded. The data is specified as a library directory or member, a SAM data set, a temporary data set, the WORK file or portions thereof, the volume table of contents of a specific disk volume, or the collective temporary data set names. The member of a library or SAM data set may be reformatted by a user written subroutine before being listed on the OMQ. Only one LIST is allowed per EDIT entry.

/LIST Function-parameter [LIBRARY=name] [IMMED]

LIST Function	Function-Parameter	LIB-RARY	LIST IMMED	FORM
List of member names	LIBRARY=name	X	X	
NIPS Component member	COMPONENT=FM	X	X	
	FS			
	PASP			
	OP			
	QUIP			
Contents of a member	MEMBER=name	X	X	X
Contents of SAM data set	SAM=name		X	X
Contents of temp-data set	tempname		X	
List of temporary files	ALL		X	
Portions of the WORK file	nnnn[+ii] -nnnn[+ii] -END		X	
Contents of the WORK file	WORK		X	
Contents of VTOC	VOL=devicetype=volid		X	

MOVE (input)

The MOVE (input class) command moves all or parts of a temporary data set or the Output Message Queue to a specified position in the WORK file.

```
          tempname [          -nnnn[+ii]]                [nnnn[+ii]]
/MOVE     OMQ      [nnnn[+ii] -END      ] TO WORK      [END      ]
```

MOVE (output)

The MOVE (output class) command moves all or part of the WORK file to a temporary data set that has been created by ALLOCATE or HOLD. Any previous contents of the data set are lost.

```
                    -nnnn[+ii]
/MOVE  nnnn[+ii] -END          TO tempname
```

PUT

The PUT command stores the WORK file or a temporary data set on a library. The receiving library and member names must be specified as NEW or OLD. Recording the name of the associated NIPS component and resequencing of the data are optional entries.

```
                                                                  FS
                                                                  FM
/PUT MEMBER=name [LIBRARY=name] [NEW] [COMP=RASP] [RESEQ] [WORK    ]
                                [OLD]  [OP  ]      [    tempname]
                                       [QUIP]
```

RESEQ

The RESEQ command causes the WORK file to be resequenced, and thus is only valid for sequenced WORK files.

/RESEQ

SETUP

The primary function of the SETUP command is to delete the existing WORK file and initialize a new one. Coincident options include changing the WORK file mode, changing the terminal input format, and releasing enqueued library members.

```
                [NEW     ]      [SEQ  ]          [OFF    ]
/SETUP USING=   [tempname] [NOSEQ] FORMAT=      [POOL   ] [FREE=name] [SIZE=nnnn]
                [WORK    ]                      [n,...,n]
                [OLD     ]
```

SIGNOFF

The function of the SIGNOFF command is to wrapup the terminal session. The delete option causes deletion of existing temporary data sets, and must be coded if any exist. The delete option is mandatory rather than default to make the user cognizant of this action.

/SIGNOFF [DELETE]

SIGNON

The SIGNON command initializes the terminal session and therefore must be the first command entered. It establishes the mode and format, and can be followed immediately by terminal input data.

```
          SEQ      [          OFF      ]
/SIGNON   NOSEQ    [FORMAT=   POOL     ]
                   [          n,...,n  ]
```

SUBMIT

The SUBMIT command indicates that the WORK file is a complete batch job and that it should be entered into the input stream for batch processing. In addition, this command deletes the WORK file, unless the SAVEWORK parameter has been entered.

/SUBMIT [SAVEWORK]

UTIL

The UTIL command performs certain specified OS-type utility functions.

```
          LOCATE    DSNAME=name
          CATLG     DSNAME=name [VOLUME=unit=volid]
/U        UNCATLG   DSNAME=name
          SCRATCH   DSNAME=name [VOLUME=unit=volid][PURGE]
          RENAME    DSNAME=name NEWNAME=name[VOLUME=unit=volid]
```

VERIFY

The VERIFY command passes the WORK file to the specified NIPS component or assembler for diagnostic analysis. The FILE parameter describes the pertinent FFT. The LIBRARY option is coded if required by the NIPS component diagnostic

routine. The FILE, VOL, and LIBRARY parameters are not required for verification of ALC source code.

```
            FS
            FM
   /VERIFY  RASP  FILE=name  [VOL=volid] [LIBRARY=name]
            OP
            QUIP
            ASM
```

5.6 Sequence Matrix for EDIT Component Commands

Command	Valid Next Statement: GET	SETUP	SIGNOFF	any command other than SIGNON	must be last command in an entry
ALLOCATE				X	
CHANGE				X	
DELETE				X	
FORMAT				X	
GET				X	
HOLD	X	X			
MOVE				X	
PUT				X	
RESEQ				X	
SETUP				X	
SIGNOFF					X
SIGNON				X	
SUBMIT			X		
UTIL				X	
VERIFY					X

5.7 Terminal Response Messages

This section describes those messages received at a terminal during an EDIT session. Following the entry of the ¬E statement, the following message is displayed at the terminal by the TP Monitor:

EOM RECEIVED.

For an initial entry to EDIT, the following message is also displayed on the line immediately below EOM RECEIVED.

EDIT STARTED.

Progress messages are displayed at various points throughout the EDIT processing to keep the user informed of his status. After all command statements have been interpreted for valid options, one of the following messages is displayed.

a. If any errors have been found, they have been listed on the EMQ and the message is:

   ERROR(S) FOUND IN COMMANDS. OUTPUT ON EMQ.
   START CONVERSATION.

b. After EDIT has completed execution of the command functions, one of the following messages is displayed:

   - If a critical error occurs to prevent successful execution of the functions, processing terminates at that point with the following message:

     UNABLE TO CCNTINUE PPOCESSING EDIT ENTRY.
     DISPLAY EMQ.

   - If no errors have occurred, the normal completion message is:

     EDIT PROCESS COMPLETED. START CONVERSATION.

     The audit trail resides on the EMQ and any LIST output resides on the OMQ.

- If the SIGNOFF command has been entered to conclude the terminal session, the message is:

  TERMINAL SIGNED OFF FROM EDIT.

- If the VERIFY command has been entered to execute the NIPS component source language edit, the message displayed is:

  EDIT PROCESS COMPLETED. COEDIT INITIATED.

  When verification actually begins, the following message is displayed:

```
FS
FM
OP    LANGUAGE VERIFICATION PROCESSOR HAS STARTED.
RASP
QUIP
ASM
```

  At the completion of the source language edit, COEDIT displays the following message:

```
FM
FS
OP   LANGUAGE CHECK COMPLETE.  NO ERRORS.
RASP  OUTPUT ON EMQ.
QUIP
ASM
```

  Diagnostic messages from the component reside on the EMQ.

5.8 Terminal Session Example

The following is an example of an entire terminal session utilizing the TEST360 file in which the user will:

a. Generate and error-scan a new range logic statement.

b. Correct and, again, error-scan the logic statement.

c. Temporarily save the logic statement and retrieve and modify another logic statement.

d. Temporarily save the second logic statement and retrieve, modify, and list JCL to submit an FM job.

e. Save the two logic statements on the library and submit a job to compile the two logic statements.

5.8.1 Generate and Error-Scan a Logic Statement

In his initial entry into the EDIT component, the user desires to generate and error-scan a complete new logic statement. This range logic statement, CU under report SAMPLE accepts as input an 80-character transaction containing an old Short Unit Name (OLDUNIT) in columns 1-12, a new Short Unit Name (NEWUNIT) in columns 21-32, and a new Long Unit Name (UNAME) in columns 41-67. For file records in which the OLDUNIT transaction field matches the file Short Unit Name (UNIT), the file fields UNIT and UNAME are to be updated with the information in the transaction fields NEWUNIT and UNAME. The following statements would be entered on the Input Message Queue (IMQ) to generate and error-scan the logic statement. The numbers to the right of the statements would not be entered by the user but have been added to provide line references for the following discussion.

```
/SIGNON                                                    1
$FMS/COM,TEST360                                           2
$ASP,SAMPLE,CU,80                                          3
$OLDUNIT,1,12                                              4
$NEWUNIT,21,32                                             5
/F    POOL                                                 6
   POOL                                                    7
   COA  $OLDUNIT,UNIT                                      8
   BNE    END                                              9
   NAL  $NEWUNIT,UNIT                                      10
   MAL  $UNAME,UNAME                                       11
END  HLT                                                   12
   END                                                     13
/V   FM FILE=TEST360  LIB=TEST360L                         14
¬E                                                         15
```

The SIGNON command, statement 1, serves two functions. First, it initiates the entire terminal session. Second, it establishes the mode of the WORK file and the format of the terminal input. Since the SEQ/NOSEQ parameter of the SIGNON command was allowed to default to SEQ, the terminal input lines supplied by the user are numbered as they are placed in the WORK file. Further, since the FORMAT parameter was also allowed to default to OFF, the first four lines of terminal input, statements 2-5 are placed in the WORK file without reformatting. Statement 6 is a FORMAT command with the POOL option and overrides the default OFF option of the SIGNON command. Consequently, statements 7-13 are reformatted using POOL language conventions as they are placed on the WORK file.

Statement 14, the VERIFY command, indicates that the WORK file created by the previous commands is to be passed to the NIPS FM component for diagnostic scanning.

The final statement, number 15, consisting of logical-not (¬)E indicates to the TP monitor that the EDIT component is to be executed with the previous statements as input.

Upon completion of processing, the WORK file would contain the following lines.

```
                        1    2                          7       8
columns  1    6         6    1                          3       0
         $FMS/COM,TEST360                                00000001
         $ASP,SAMPLE,CU,80                               00000002
         $OLDUNIT,1,12                                   00000003
         $NEWUNIT,21,32                                  00000004
                        POOL                             00000005
                        COA  $OLDUNIT,UNIT               00000006
                        BNE  END                         00000007
                        NAL  $NEWUNIT,UNIT               00000008
                        MAL  $UNAME,UNAME                00000009
                   END  HLT                              00000010
                        END                              00000011
```

The user would be informed that the NIPS FM component had detected two errors in the generated logic statement. First, WORK file line number 8 contains an unrecognizable operational code since NAL was entered instead of the correct MAL. Second, WORK file line number 9 contains an undefined transaction field name since UNAME was omitted from the TDD of the logic statement. These errors are corrected in the next entry to EDIT.

5.8.2 Correct and Error Scan the Logic Statement

In this entry to EDIT, modifications to correct the errors detected in the logic statement generated in the previous entry will be made and then the logic statement will again be scanned for errors. The following statements are entered on the IMQ. Again, the numbers to the right of the statements are for reference only and would not be entered.

```
/F                                                       1
/I     4                                                 2
$UNAME,41,67                                             3
/C     8     'NAL'MAL'                                   4
/V     FM    FILE=TEST360                                5
¬E                                                       6
```

Statement 1 is a FORMAT command defaulting to the OFF option. This is necessary since a POOL format is in effect from the previous entry, and a line in non-POOL format is to be inserted in the WORK file.

Statement 2 is an INSERT command indicating that the following line of terminal input is to be inserted in the WORK file following the fourth WORK file line. Statement 3 is a TDD record defining the transaction field UNAME which was omitted from the original TDD. This corrects the second error detected in the previous FM error-scan.

The first error detected in the previous FM error scan is corrected by statement 4, a CHANGE command with the modification option. This statement indicates to the EDIT component that in line number 8 of the WORK file, any occurrence of the characters NAL is to be replaced by the characters MAL.

Statement 5, the VERIFY command, again indicates that the WORK file is to be passed to the NIPS FM component for diagnostic scanning.

The final statement causes execution of the EDIT component to process the previous commands.

When processing is complete, the WORK file contains the following lines.

```
                        1       2                              7       8
column1     6           6       1                              3       0
      $FMS/COM,TEST360                                         00000001
      $ASP,SAMPLE,CU,80                                        00000002
      $OLDUNIT,1,12                                            00000003
      $NEWUNIT,21,32                                           00000004
      $UNAME,41,67                                             00000004
                        POOL                                   00000005
                        COA     $OLDUNIT                       00000006
                        BNE     END                            00000007
                        MAL     $NEWUNIT,UNIT                  00000008
                        MAL     $UNAME,UNAME                   00000009
           END          HLT                                    00000010
                        END                                    00000011
```

Notice that the inserted TDD record is given the line number specified on the INSERT command causing two WOPK file lines to contain line number 4.

This time, the NIPS FM component diagnostic scan indicates that no errors were detected in the logic statement.

5.8.3 Retrieve and Modify a Second Logic Statement

In this entry to EDIT, the logic statement generated in the previous entries is saved in a temporary data set named LS1 and a second logic statement is retrieved from the TEST360L library, modified, and error-scanned. The modifications consist of adding the definition of a new field to the TDD of the logic statement and adding the move instruction that updates the file record from that transaction field. The lines adjacent to the inserted ones are listed so that the user may check that the inserts are correctly placed for his purposes. The statements placed on the IMQ are as follows:

```
/H    LS1                                          1
/G    MEM=SAMPUO   UPDATE                          2
/I    9                                            3
$OPCON,67,72                                       4
/L    9-10                                         5
/I    23                                           6
/F    POOL                                         7
      MAL  $OPCON,OPCON                            8
/L    22-25                                        9
/V    FM   FILE=TEST360                            10
¬E                                                 11
```

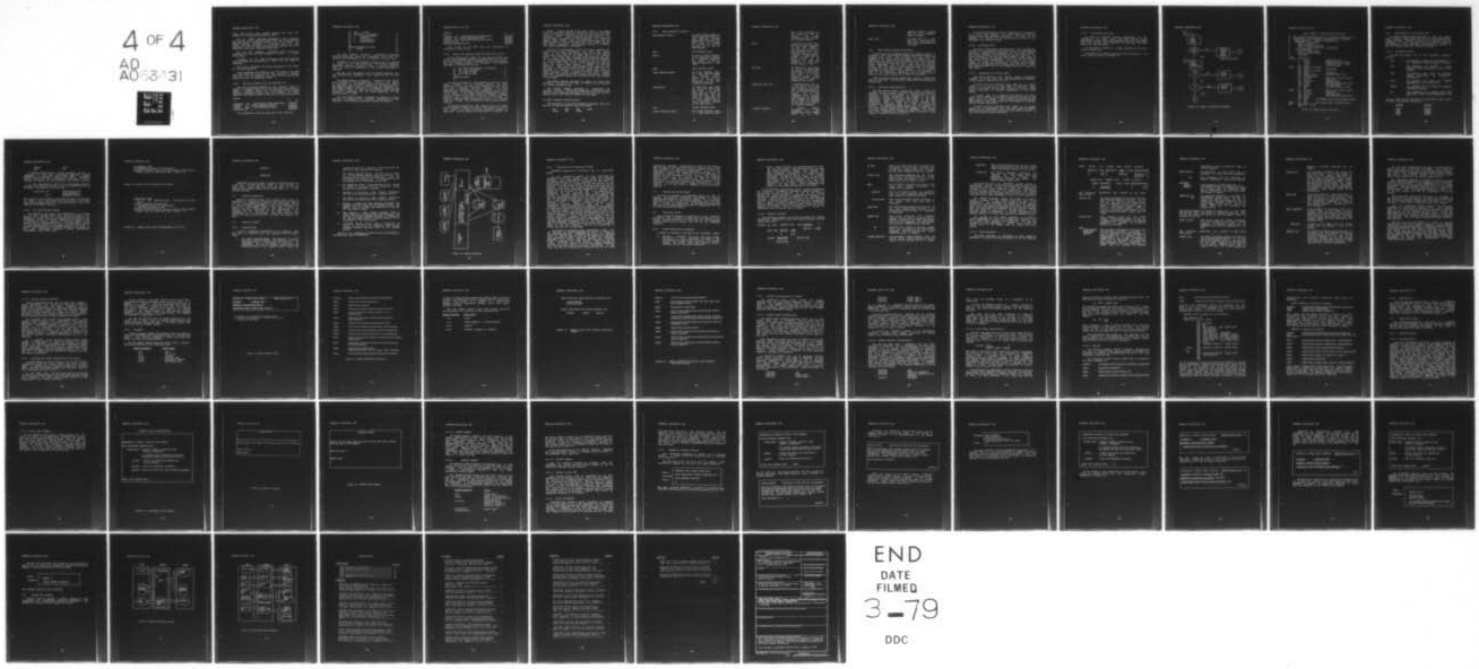
Statement 1, the HOLD command, creates a temporary data set named LS1 and stores the first logic statement in it.

The second statement, a GET command, retrieves the second logic statement from member SAMPUO of the TEST360L library. The library parameter is not specified since the last library specified was TEST360L and, consequently, that is the default library. The lines in the member are assumed to contain line numbers since NOSEQ was not specified. Since the logic statement is to be stored back on the library later, the UPDATE option is specified.

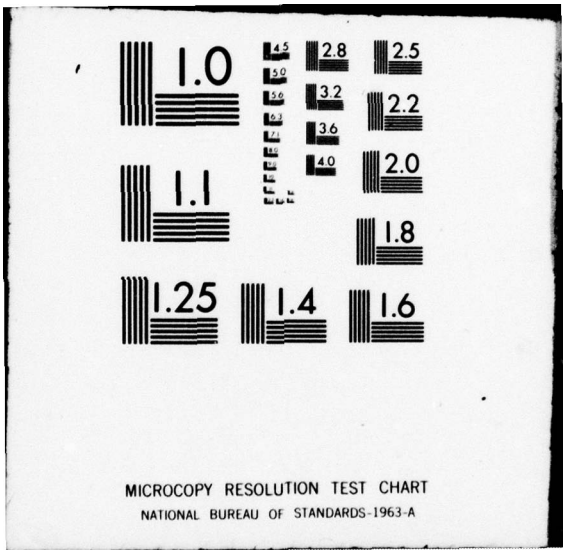
The modification commands following the GET command reference the retrieval member since there is no input WORK

AD-A063 431 COMMAND AND CONTROL TECHNICAL CENTER WASHINGTON D C F/G 9/2
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)
SEP 78 C K HILL
UNCLASSIFIED CCTC-CSM-UM-15-78-VOL-6 SBIE-AD-E100 131 NL

4 OF 4
AD
AD A063131



END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

file. (The initial HOLD command deleted the WORK file created on the previous entries to EDIT.)

The two INSERT commands, statements 3 and 6, indicate the location in the logic statement where the TDD record and move instruction, statements 4 and 8, are to be placed. The FORMAT command, statement 7, indicates that the move instruction (and any later terminal input records) are to be formatted using POOL language conventions.

The two LIST commands, statements 5 and 9, cause the inserted lines and the lines adjacent to them to be placed on the OMQ.

Statement 10, the VERIFY command, causes the modified logic statement to be scanned for errors by the NIPS FM component.

The final statement initiates processing of this entry by the EDIT component.

Upon completion of processing, the OMQ contains the WORK file lines 9-10 and 22-25 and the EMQ contains a message from the NIPS FM component indicating that no errors were detected during the diagnostic scan.

5.8.4 Retrieve, Modify, and List Required JCL

The logic statement modified in the previous entry is saved in a temporary data set (LS2); and the JCL required to submit a library update run is retrieved from a library (JCLLIB), modified for this run, and listed. The FMJCL member of the JCLLIB library contains the following records.

```
Column:
                                                        7      8
1                                                       3      0
//JOBNAME   JOB   (ACCOUNTING),PGMR,TYPRUN=HOLD        00000001
//JOBLIB    DD    DSN=FFS.JOBLIB,DISP=SHR             00000002
//STEP1     EXEC     XFM,ISAM=FILE,LIB=PDS            00000003
//FM.SYSIN  DD    *                                   00000004
```

The statements entered by the user on the IMQ are:

```
/H   LS2      RESEQ                                  1
/G   MEM=FMJCI    LIB=JCLLIB                        2
/C   1     'JOBNAME'LIBUPD'                         3
/C   1     'PGMR'DOE'                               4
/C   3     'FILE'TEST360'                           5
/C   3     'PDS'TEST360'                            6
/F                                                  7
/I                                                  8
$FMS/LIB,TEST360,LS,DISK                            9
/L       IMMED                                      10
¬E                                                  11
```

The HOLD command, statement 1, generates a temporary data set named LS2, places the renumbered WORK file (the second logic statement) in it, and deletes the WORK file.

Statement 2, the GET command, retrieves the JCL required to submit an FM job from JCLLIB. The library parameters specifying JCLLIB are required since a library different from the last one referenced in this session (TEST360L) is required.

The next four statements (3-6) are CHANGE commands with the modification parameters that tailor the JCL for this job.

The INSERT command, statement 8, inserts the FMS control card, statement 9, in the WORK file. Since no line number is specified on the INSERT command, it defaults to END so that the FMS control card is placed after the retrieved JCL. The FORMAT command, statement 7, is required preceding the FMS control card to turn off the automatic POOL language convention formatting that had been specified in the previous entry to EDIT.

The LIST IMMED command, statement 10, causes the entire WORK file as modified to be displayed immediately on the terminal. The modified lines would appear as follows:

```
columns                                                    7      8
1                                                          3      0
//LIBUPD   JOB   (ACCOUNTING),DOE,TYPRUN=HOLD             00000001
//JOBLIB   DD    DSN=PFS.JOBLIB,DISP=SHR                   00000002
//STEP1    EXEC  XFM,ISAM=TEST360,LIB=TEST360              00000003
//FM.SYSIN  DD   *                                        00000004
$FMS/LIB,TEST360,LS,DISK                                   00000005
```

This would be the WORK file upon completion of processing of this entry.

5.8.5 Update the Library, Submit the Job, and Sign Off

In the final entry to EDIT, the two logic statements are stored on the TEST360L library and placed after the FM JCL on the WORK file, the job is submitted for batch processing, and the terminal session is terminated. The following statements are placed on the IMQ.

```
/P   LS1 COMP=FM MEM=SAMPCU  LIB=TEST360L      1
/P   LS2  OLD  MEM=SAMPUO                      2
/M   LS1 2-END TO WORK                         3
/M   LS2 2-END TO WORK                         4
/SUB                                           5
/SIGNOFF DELETE                                6
¬E                                             7
```

The first statement is a PUT command to store the first logic statement on the TEST360L library. The first parameter, LS1, is the name of the temporary data set where the logic statement currently resides. The second parameter, FM, specifies the NIPS component of the data being stored. The component must be specified since this is a new member. The member parameter specifies the member name to be given to the data, and the library parameter specifies the library where the data is to be stored. The library, TEST360L, must be specified since it is different from the library last referenced in this terminal session (JCLLIB).

The second statement is a PUT command for the second logic statement. Again, the name of the temporary data set, LS2, where the logic statement currently resides is specified. The OLD parameter is specified to indicate that

a member is being updated rather than created. The member could not have been updated if UPDATE had not been specified on the GET command (statement 2) in the third entry to EDIT (section 5.8.3). Since the member is being updated, it is not necessary to specify the NIPS component. The specified member name, SAMPUO, must match the member name that had been specified in the GET command with the UPDATE option. The library parameter was allowed to default to TEST360L, the library specified in statement 1.

The next two statements, 3 and 4, are MOVE commands that place the two logic statements, less their FMS control cards, in the WORK file following the JCL and FMS control cards that were placed in the WORK file in the previous entry. The first logic statement from temporary data set, LS1, is placed right after the FMS control card by statement 3. It is placed after the FMS control card (at the end of the input WORK file) since no line number was specified after the keyword WORK causing a default to the END parameter. The second logic statement from temporary data set, LS2, is placed at the end of the input WORK file by statement 4. However, it is also placed after the first logic statement since execution of the MOVE command (statement 3), placing the first logic statement in the WORK file, is completed before the execution of the next command is initiated.

The SUBMIT command, statement 5, causes the WORK file containing the JCL and two logic statements to be submitted as a batch processing job.

The SIGNOFF command, statement 6, terminates the terminal session. The DELETE parameter is specified since there are two temporary data sets, LS1 and LS2, that must be deleted before the terminal session may be ended.

5.9 EDIT Component Reserved Words

The following are the EDIT component reserved words and may not be used as temporary data set names.

ALL	NEW	OMQ	WORK
IMMED	OLD	RESEQ	

5.10 EDIT Component Glossary

Edit Message Queue - The Edit Message Queue is a disk data set where the EDIT component command diagnostic messages and audit trail are placed for access by the user through use of the TP PAGE routine.

EMQ - Edit Message Queue

Entry - An entry to EDIT consists of all statements entered at a terminal preceding the entry of the TP command logical-not EDIT (¬E).

IMQ- Input Message Queue

Input Message Queue - The Input Message Queue is a disk data set where the terminal input, making up one entry to EDIT, is stored for processing by the EDIT component routines.

NOSEQ Mode - The WORK file may exist in one of two modes, SEQ and NOSEQ. In NOSEQ mode, all 80 bytes of each WORK file line are used for data and the lines are referenced by a line counter giving their relative position in the WORK file.

OMQ - Output Message Queue

Output Message Queue - The Output Message Queue is a disk data set where

data to be displayed to the terminal user is stored. The user accesses the data through the TP PAGE routine.

Pass - A pass at the WORK file consists of one execution of the EDIT component routines that create a new updated WORK file. Normally there is a one-for-one correspondence between a pass and an entry though it is possible to have two passes at the WORK file in one entry to EDIT.

SEQ Mode - The WORK file may exist in one of two modes, SEQ and NOSEQ. In SEQ mode, bytes 73-80 of the 80 byte WORK file lines actually contain a line number which is used to reference the lines.

Temporary Data Set - A temporary data set is a disk data set in the same format as the WORK file in which portions or all of the WORK file are stored. Up to five temporary data sets per terminal may be created upon command of the user but all must be deleted by the user by the end of the terminal session.

Terminal Session - A terminal session consists of all processing done between entry of an EDIT

component SIGNON command and an EDIT component SIGNOFF command.

WORK file - The WORK file is a disk data set containing the data being manipulated by the user.

5.11 User Written Format Subroutine

The GET and LIST commands contain the linkage needed to interface with a user written format subroutine designed to process input data. This provides the user with the capability of displaying data lines on a terminal that are wider than the screen, since extraneous data can be stripped off or displayed on the next line. Also, the user written format subroutine can be used to edit input transactions online. The user written format subroutine may be as simple or complex as the application requires. If the keyword FORM=name is specified on either the GET or LIST command, an exit to the specified subroutine will be made prior to writting the input records on the WORK or OMQ files.

When writing the format subroutine certain conventions must be followed. The following subsections described such conventions.

5.11.1 Interface Specifications

The user written format subroutine should follow the standard OS/360 linkage and NIPS Subloader conventions as specified in section 3 of Volume I, Introduction to File Concepts. Three parameters are passed to the user subroutine. Parameter one is the entry point to the NIPS subroutine loader. Parameter two points to the user information (that is, input and output buffers and blocksize). Parameter three is a cell for return code storage. The subroutine loader entry point is provided to the user format subroutine so that request to load or link to other routines and/or tables is possible. The return code cell designated by parameter three or register 15 can contain a return code of 0, 4, or 8.

The format subroutine should be compiled and linkedited to a user file library. The format subroutine must be loaded by the NIPS Subloader procedure. The size of the argument and function is 12, since input and output to the subroutine are address strings.

5.11.2 Parameter Area

At entry to the format subroutine the user information pointed to by parameter two, will contain two addresses and a halfword containing the blocksize of the input buffer. The first address is the address of the input buffer. The second address is that of the output buffer. The size of the input buffer is determined by the DCB, BLKSIZE field of the data set to be reformatted. The size of the output buffer is always 80 characters. The user written format subroutine is responsible for knowing when the input buffer has been exhausted and it is necessary for EDIT to read the next block.

5.11.3 Consequence of Return Codes

Upon return from the user written format subroutine, register 15 or the return code cell designated by parameter 3 can contain a value of 0, 4, or 8.

A return code of 0 indicates that there is an output record to be written on the WORK or OMQ file. After writting the output record, EDIT returns to the user written format subroutine to continue processing of the input buffer.

A return code of 4 indicates that there is an output record to be written on the WORK or OMQ file and that a new input block is needed. After writing the output record, EDIT reads the next input block and then returns to the user written format subroutine.

A return code of 8 indicates that there is no output record to be processed, but a new input block is needed. EDIT reads the next input block and then returns control to the user written format subroutine.

5.11.4 Interfacing with EDIT

In order to maintain maximum flexibility in the reformating of input data, the user written format subroutine has been given the responsibility for unblocking the input buffer and indicating to EDIT when the input buffer has been exhausted.

An overview of linkage to a format subroutine in EDIT is given in figure 10.

An example of a format subroutine which reformats a 133 input record is given in figure 11.

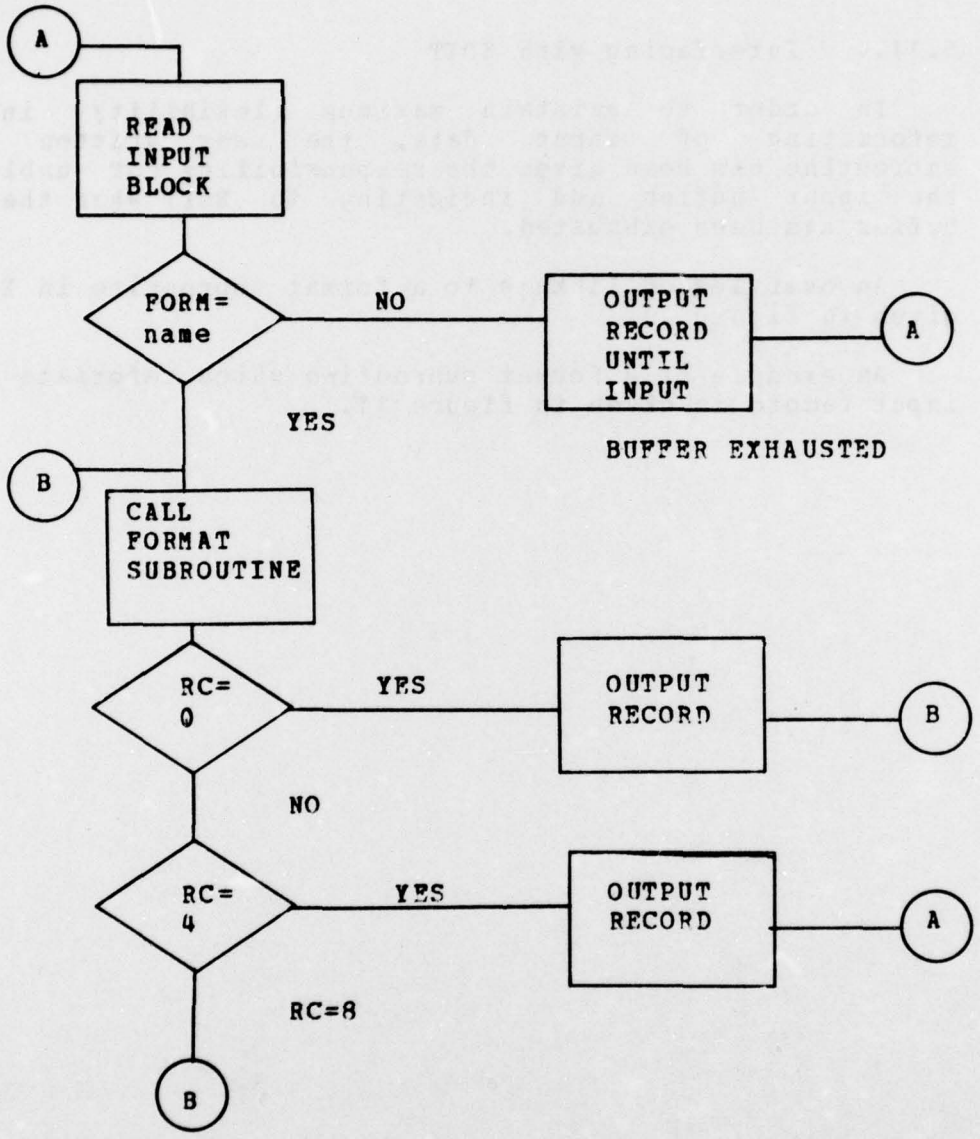


Figure 10. Format Subroutine Interface

```
                TITLE 'FORMAT - SAMPLE FORMAT SUBROUTINE'
*
*    THIS ROUTINE ILLUSTRATES HOW A 133 CHARACTER RECORD CAN BE
*    REFORMATTED TO 80 CHARACTERS FOR DISPLAY ON A TERMINAL
*      UPON ENTRY REG 1 = ADDR OF PARAMETER LIST
*      PARAMETER LIST CONTAINS
*         ADDR OF SUBSUP
*         ADDR OF USER INFORMATION
*         ADDR OF RETURN CODE CELL (1 CHARACTER)
*      USER INFORMATION
*         ADDR OF INPUT RECORD
*         ADDR OF OUTPUT BUFFER
*         BLOCKSIZE
*
FORMAT   FFSBEGIN 10
         L     1,4(1)                  PICKUP USER INFO
         L     2,0(1)                  ESTABLISH ADDR OF INPUT
         L     3,4(1)                  ESTABLISH ADDR OF OUTPUT
         LH    4,8(1)                  PICKUP BLKSIZE
AGAIN    CLC   120(4,2),=C'PAGE'
         BE    COND8                   YES-BRANCH
         CLC   120(4,2),=C'DATE'
         BE    COND8                   YES-BRANCH
         MVC   0(28,3),4(2)            REFORMAT PRINT LINE
         MVC   28(52,3),39(2)
         LA    2,133(2)                BUMP TO NEXT INPUT RECORD
         ST    2,0(1)                  SAVE ADDR
         SH    4,=H'133'               SUBTRACT LRECL
         STH   4,8(1)                  SAVE REMAINING BLOCK COUNT
         BZ    COND4                   IF ZERO-BRANCH
         SR    15,15                   SET RETURN CODE TO ZERO
         B     RET
COND8    LA    2,133(2)                BUMP TO NEXT INPUT RECORD
         ST    2,0(1)                  SAVE ADDR
         SH    4,=H'133'               SUBTRACT LRECL
         STH   4,8(1)                  SAVE REMAINING BLOCK COUNT
         BNZ   AGAIN
         LA    15,8          SET RETURN CODE TO EIGHT-IGNORE OUTPUT
         B     RET
COND4    LA    15,4          SET RETURN CODE TO FORM-NEED BUFF
RET      FFSRETRN  RC=(15)
         END
```

Figure 11. Sample Format Subroutine

5.12 Verification of ALC Source Code

The VERIFY command allows the user to pass ALC source code, stored in the current WORK file, to the assembler for syntax checking. Through the use of an auxillary control statement, the user may specify any of several options to the assembler. Upon completion, the results are placed on the Edit Message Queue for review by the user.

5.12.1 Assembler Options

The user may specify any of the following assembler options:

LIST	-	An assembler listing will be produced on the auxillary printer (ASMPRT DD card).
XREF	-	The assembler will produce a cross reference table of symbols as part of the listing.
RENT	-	The assembler will check for possible coding violations of program re-enterability.
TEST	-	The asembler will allow for TESTRAN statements in the source code.
NOALGN	-	The assembler will not check for improper boundary alignment.
DOS	-	The assembler will compile like Disk Operating System (DOS) Assemblers D and F.

When an option is not specified, the default for that option is used. The default values are:

Option	Default
LIST	NOLIST
XREF	NOXREF
RENT	NORENT
TEST	NOTEST

NOALGN	ALGN
DOS	OS

A complete description of these assembler options is contained in the IBM System/360 Operating System Assembler(F) Programmer's Guide (GC26-3756). Assembler options listed in the Programmer's Guide but not in the section are illegal and will terminate the VERIFY process.

If the user wants to use any of the available options, he must place them on the first line of the WORK file to be verified. The format of the line is:

Positions:	1-7	*ASMOPT(Required)
	8	Blank(Required)
	9-71	Assembler Options.

The options may be listed in any order and must be separated by commas. The first blank after the start of the option list will terminate the list. Any error in this line will terminate the VERIFY function.

5.12.2 ALC Verification Results

The results of ALC source code verification are placed on the Edit Message Queue for review by the user. The results are in the form of a SYSTERM listing produced by the assembler. Figure 12 contains a sample of the results from a successful verify. Figure 13 contains a sample of the results from an unsuccessful verify. The underlined number is the sequence number of the line containing the error. A complete description of the SYSTERM listing is contained in the IBM System/360 Operating System Assembler(F) Programmer's Guide (GC26-3756).

```
ASSMELBER(F) DONE
NO STATEMENTS FLAGGED IN THIS ASSEMBLY
*OPTIONS IN EFFECT* NOLIST, NODECK, NOLOAD, NORENT,NOXREF,
  NOTEST, ALGN, OS, TERM, NUM, STMT, LINECNT = 55
```

Figure 12. Output from a Successful ALC Verify

```
ASSEMBLER(F) DONE
  550   72    MUC    IORTNS(8),0(R1)    SAVE ADDR OF I/O RTNS
    A2917
IEU088  UNDEFINED OPERATION CODE
  1 STATEMENT FLAGGED IN THIS ASSEMBLY
 12 WAS HIGHEST SEVERITY CODE
*OPTIONS IN EFFECT* NOLIST, NODECK, NOLOAD, NORENT, NOXREF,
  NOTEST, ALGN, OS, TERM, NUM, STMT, LINECNT = 55
```

Figure 13. Sample Output from an Unsuccessful ALC Verify

SECTION 6

FORMATTER

FORMATTER allows the user to define a display format for a CRT and to use this display format at the terminal to enter data. This data can then be used by the NIPS application programs FMSODA, QUIP, BLAST, EDIT, or a user written program.

6.1 General Description

Defining a display format for on-line data entry is analogous to designing a form and then using that form to fill in the requested information. Once the display format or form has been established, it can be used repeatedly to enter data. The display format, much like a form, can contain specific indications of the information to be supplied and the locations on the format where that information is to be entered. The data entered on the format can then be extracted and arranged as desired for use.

6.2 General Concepts

6.2.1 Related Terms

In order to understand FORMATTER, it is important that the user is familiar with the meaning of the following terms:

a. IMQ (Input Message Queue) - the data set on which terminal input is placed. See subsection 2.3.1 of this manual for a discussion of the IMQ. In FORMATTER, the IMQ is created according to the specifications of the display format using the

operator-keyed data. The IMQ is then available for use by the NIPS application program.

b. OMQ (Output Message Queue) - the data set on which all TP application program output is written. The contents of the OMQ can be displayed by using the conversational program PAGE. See subsection 2.4 of this manual for a discussion of the OMQ.

c. One FORMATTER cycle - the processing of one format or chain of formats and the execution of the invoked NIPS application program.

d. Skeleton - an entry in a user library containing the CRT field descriptions for a display format.

e. IMQ table - an entry in a user library containing the IMQ field definitions for a display format.

f. Format - a member of a user library containing the skeleton and IMQ table for one display format. The member name is the same name specified by the user for the display format.

g. User library - a file library normally used to store compiled, executable NIPS user programs. See section 10 "Source Language Storage," in Volume VII of the NIPS Users Manual, "Utility Support," for a discussion of the file library.

h. FORMATTER display action code - indicates the action to be taken by FORMATTER after the CRT operator finishes entering data on the display format.

Figure 14 is a schematic overview of the relationship of the data sets used by FORMATTER.

TERMINAL PROCESSING (TP)

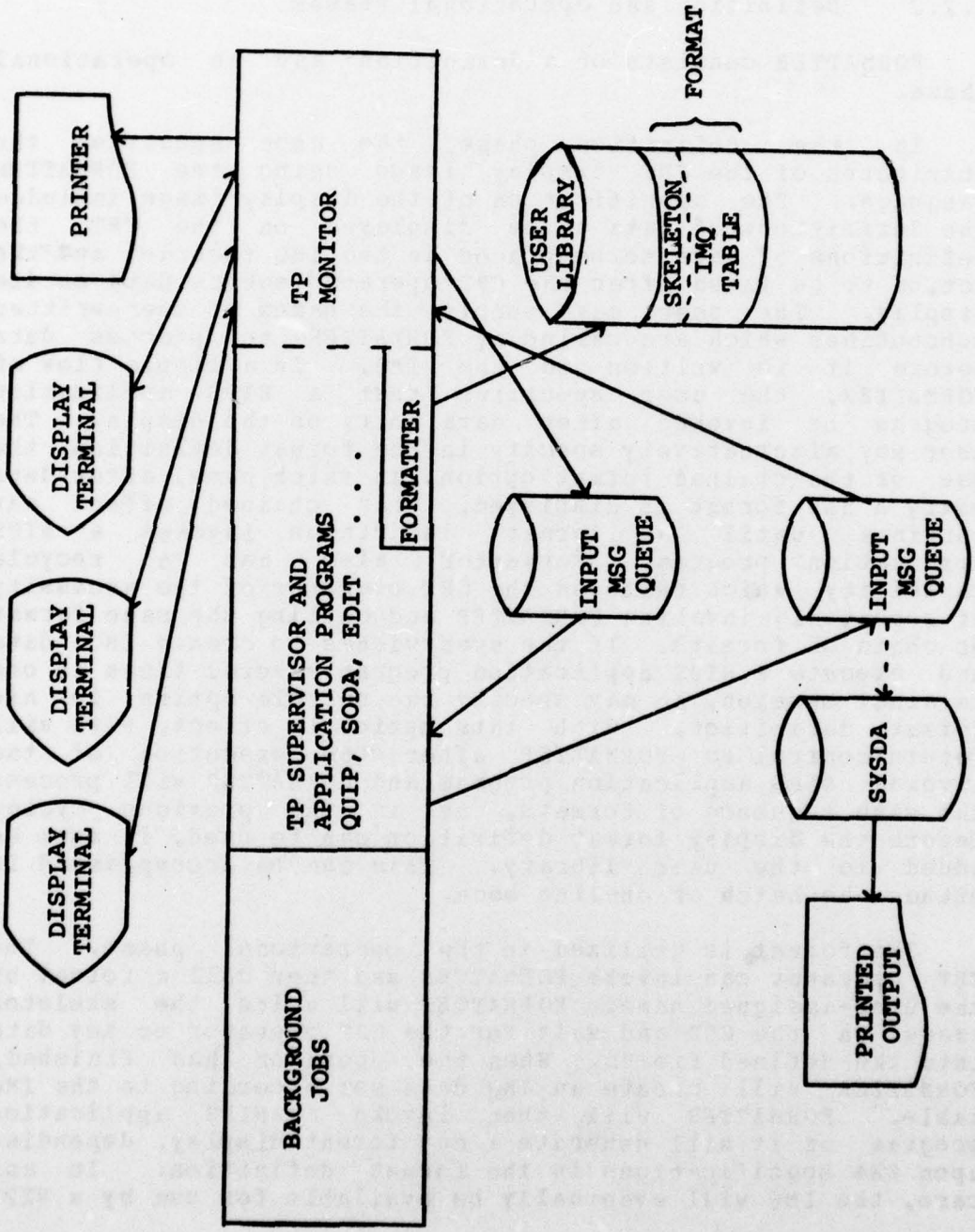


Figure 14. System Overview

6.2.2 Definition and Operational Phases

FORMATTER consists of a definition and an operational phase.

In the definition phase, the user specifies the attributes of the CRT display image using the FORMATTER language. The specification of the display image includes the definitions of data to be displayed on the CRT, the definitions of data to be placed in the IMQ records, and the action to be taken after the CRT operator enters data on the display. The user may supply the names of user-written subroutines which are called by FORMATTER to process data before it is written to the IMQ. In a simple flow of FORMATTER, the user specifies that a NIPS application program be invoked after data entry on the display. The user may alternatively specify in the format definition the use of the chained format option, in which case, after data entry a new format is displayed. This chained effect can continue until a format definition invokes a NIPS application program. Formatter also has a recycle capability which relieves the CRT operator of the necessity of repeatedly invoking FORMATTER and calling the same format or chain of formats. If the user wishes to create IMQ data and execute a NIPS application program several times in one terminal session, he may specify the recycle option in his format definition. With this option in effect, NIPS will return control to FORMATTER after the execution of the invoked NIPS application program and FORMATTER will process the same sequence of formats, as in the previous cycle. Before the display format definition can be used, it must be added to the user library. This can be accomplished in either the batch or on-line mode.

The format is utilized in the operational phase. The CRT operator can invoke FORMATTER and then call a format by the user-assigned name. FORMATTER will write the skeleton image on the CRT and wait for the CRT operator to key data into the defined fields. When the operator has finished, FORMATTER will create an IMQ data set according to the IMQ table. FORMATTER will then invoke a NIPS application program or it will generate a new format display, depending upon the specifications in the format definition. In any case, the IMQ will eventually be available for use by a NIPS

application program. While keying in data on the display, the CRT operator may also override the display action code. This enables the operator to pass control to a format not specified in the current format definition, invoke a NIPS component, prevent the execution of a NIPS component, or scratch IMQ data created up to that point. This override ability allows the CRT operator to break the FORMATTER recycle loop. Aside from the actual use of a format definition, the user may perform two other functions at the terminal. The user may place a format definition on a user library, or he may issue a command to end the terminal session.

6.3 Format Controlled Access

The NIPS TP component and FORMATTER permit the user to specify user-written validation subroutines to control unauthorized access to the display formats. If a validation subroutine is used, an exit to the subroutine will be made when a format is referenced. All validation subroutines must follow the conventions described in subsection 7.14 of the NIPS Installation Manual.

6.4 Definition Phase

This subsection presents a description of the FORMATTER statements used to define a format and a discussion of how to add a format definition to a user library. The following example should help to clarify the use of the FORMATTER language.

6.4.1 Format Definition Statements

A format is defined by the use of two statement types.

a. DISPLAY - specified one time for each format definition. This statement identifies the associated NIPS application program, the display device, the format chain and automatic recycle options, and the IMQ record reference.

b. FIELD - used to define the characteristics of both CRT and IMQ fields. These characteristics include the field's location, length, data content, and user subroutine relationship. In addition, the FIELD statement can include a CRT field's keying potential, intensity, and data type. In one FIELD statement, the user can define one CRT field or one IMQ field, or he can combine CRT and IMQ definitions if the data in the CRT field is to be moved to the IMQ field. FIELD statements may appear in the format definition in any order.

In the statement descriptions below, several conventions are followed. Parameters enclosed in square brackets [ ] are optional and may be omitted. If one of the options within the square brackets is completely underlined, it is the default option and will be assumed if the parameter is omitted. Capitalized options are keyword and must be coded as shown; lowercase options are replaced by the appropriate value. Where a keyword parameter is partially underlined, only the underlined portion must be coded, though the entire keyword may be coded. Where a vertical list of options is specified, only one option may be chosen.

6.4.1.1 DISPLAY Statement

The DISPLAY statement is the first statement in a format definition. There may be only one DISPLAY statement for any format definition.

```
DISPLAY  ID  name  [DEVTYP code]  NEXT [EXECUTE          ]
                                       [format - name    ]
         COMP name  [REPEAT  YES]
                    [        NO ]
         IMQREF  [ABSOLUTE]        [CURSOR nn]
                 [RELATIVE]
```

ID name	This is the name that will be used in all references to this format. The name may be from one to eight characters in length and the first character must be a letter.
DEVTYP code	This parameter indicates the CRT device type and is for future use only. At the present time, the default and the only value which may be coded is 3270.
NEXT	This parameter specifies the action to be taken after format processing is completed by FORMATTER.
EXECUTE	This is the default value and indicates that the NIPS component defined by the COMP parameter should be executed.
format-name	This entry indicates format chaining; a new format with this name will be processed.
COMP name	The COMP parameter specifies the name of the NIPS application program to be executed after this format is processed by FORMATTER.
REPEAT YES	This value indicates the automatic recycle option, and is pertinent only if EXECUTE is coded in the NEXT parameter. After the NIPS component has been invoked and executed, FORMATTER will recycle.
NO	FORMATTER will remain in the wait state, after the execution of the invoked NIPS component. The CRT operator must invoke FORMATTER again, if desired.
IMQREF ABSOLUTE	This parameter value indicates that the IMQLOC record number references will be related to the IMQ data set as a whole.

RELATIVE	This value indicates that the IMQ record number references will be related to only those IMQ records created in this format.
CURSOR nn	The CURSOR parameter specifies the starting CRT line number where any messages from the invoked NIPS application program will appear. The CURSOR default value is 01.

Note that the use of the FORMATTER recycle option is effectively limited to the NIPS application program, SODA, because SODA does not invoke any other NIPS application programs (i.e., PAGE) before it terminates. (See section 6.4.4.3 for QUIP exception.) Any prints to the CRT screen from the FMSODA logic statement will interrupt the automatic recycling (i.e., error messages or use of the PT2 operator).

The IMQREF parameter allows one format to be used to create a number of IMQ records with the same field definitions. By choosing the RELATIVE option and by chaining the format to itself, the format will be repeated until the CRT operator overrides the display action code. Each time the format is processed, the IMQ records created will be added to the end of the IMQ data set. If the RELATIVE option is specified in a format definition, all formats chained to it must also specify the RELATIVE option.

The purpose of the CURSOR parameter is to keep the format displayed on the CRT at the same time messages from the invoked NIPS program are displayed. The user can accomplish this by specifying a CURSOR value beyond the format area, if the format does not occupy the entire screen.

6.4.1.2 FIELD Statement

The FIELD statement is specified as many times as necessary to define the various elements of the CRT display.

```
FIELD    CRTLOC   rrcc   CRTLEN   nnnn   [VALUE   literal]

        [HIGHLITE  NO  ]  [ATTRIBUTE  ALPHA  ]  TYPE  [UNPROTECTED]
        [          YES ]  [           NUMERIC]        [PROTECTED  ]
                                                      [LITEPEN    ]

          IMQLOC nnpp   IMQLEN   nnnn    [VALUE   literal]

             [DATA  REQUIRED]        [EDIT  name]  [UNDERLINE NO ]
             [      OPTIONAL]                      [          YES]
```

The following parameters are related to CRT field definitions:

CRTLOC rrcc — The CRTLOC parameter defines the location where the CRT field is to begin. rr is the CRT row, and cc is the column where the first character in the field is to appear. Leading zeros must be coded in the row and column values. (Row 1, column 5 would be coded as 0105.) Note that multiple row-column values may be coded.

CRTLEN nnnn — This parameter gives the CRT field length. The maximum length is 1290. If a VALUE literal is coded and the CRTLEN parameter is omitted, the literal's length will be used.

TYPE

UNPROCTECTED
PROTECTED
LITEPEN — This parameter indicates if the user may key data into the field. The default will be PROTECTED if a VALUE parameter is coded; otherwise, the default will be UNPROTECTED. If a VALUE parameter is coded, and the user wishes to designate the field as lightpen selectable, the operand LITEPEN must be specified. A field that is defined as LITEPEN

selectable is also a protected field on the CRT display.

HIGHLITE YES — The characters in this field will be displayed at high intensity brightness.

NO — The characters will be displayed at normal brightness. This is the default.

ATTRIBUTE
ALPHA
NUMERIC — This parameter is applicable only to UNPROTECTED fields. If NUMERIC is coded, the operator will be prevented from keying alphabetic data into this field. The default is ALPHA.

UNDERLINE YES
NO — The UNDERLINE parameter is pertinent for UNPROTECTED fields only. If YES is coded, the field will appear with the '_' (underline) character filled in for the entire length of the field. If NO is coded, blanks will be displayed. The default is NO.

The VALUE parameter is related to either CRT or IMQ field definitions when they appear in separate statements. When CRT and IMQ fields are defined in one statement, VALUE is related to the CRT.

VALUE literal — The literal shows the actual characters that are to appear in this field. If the parameter is omitted, blanks will be displayed on the CRT or generated on the IMQ.

The following parameters are related to IMQ field defintions:

IMQLOC nnpp — The IMQLOC parameter defines the location where the IMQ field begins. nn is the IMQ record number, and pp is the position of the first character of the field in the IMQ record. Leading zeros must be coded for both the record number and

position. Multiple locations may be coded.

IMQLEN nnnn — This parameter specifies the length of the IMQ field. If a VALUE literal is coded and the IMQLEN parameter is omitted, the literal length will be used. If the CRTLEN parameter is coded, IMQLEN may be omitted if they are equal. If the specified IMQ length exceeds the number of positions in the current IMQ record, additional IMQ records will automatically be created. (IMQ record length is 80 characters.)

EDIT name — This is the name of a user subroutine or table which is to be given control before data is stored in the IMQ field. The user subroutine or table must conform to the conventions described in the NIPS User Manual, Volume I, "Introduction to File Concepts."

DATA REQUIRED — This parameter is pertinent only if a literal VALUE is not coded. An error procedure will be executed if data is not keyed in on the CRT. The error procedure will redisplay the format with any data already keyed and with asterisks in the REQUIRED field.

OPTIONAL — If data is not keyed into the field, blanks will be stored. This is the default.

WORKLOC nnn — The WORKLOC parameter allows the user to transfer data in a 500 byte TP workarea to the IMQ and/or the CRT. nnn is the starting location of the data in the workarea. Leading zeros are not required for workarea starting location. (See section 6.4.4 for more information.)

CRT and IMQ parameters may be used to define multiple CRT or IMQ fields in the FIELD statement. The CRTLOC or IMQLOC parameter may be coded with multiple location values. All other coded parameters in the statement will apply to each field. The maximum number of multiple field definitions permitted in one FIELD statement is 10.

The UNDERLINE parameter is intended to provide the user a visual means of determining the position of an unprotected field on the CRT screen. If the UNDERLINE option is coded, the entire field will appear filled with the '_' (underline) character. The user may key his data into the field in the normal manner. The '_' (underline) characters appear only on the CRT screen and are not transferred to the IMQ records.

Each separately defined CRT field must be preceded by one space on the CRT screen. In other words, between any two CRT FIELD statement definitions, there must be at least one space. This space is required by the 3270 terminal.

Light pen selectable fields appear on the CRT screen preceded by a question mark character (?). This question mark (?) is the result of the operand LITEPEN on the TYPE operator and is not defined by the user. During the operational phase when the CRT operator selects a field with the lightpen, the question mark (?) changes to a greater-than sign (>). Thus the user can easily see which fields he has selected.

Care must be taken when defining a field as lightpen selectable. The 3270 terminal requires certain minimum screen spacing before and after a lightpen selectable field. A lightpen selectable field must be preceded by at least four spaces on the CRT screen except when CRTLOC equals rr03. Three spaces must follow each lightpen field. The entire lightpen field, excluding the last three spaces, must be defined on one CRT line. This spacing is assumed by FORMATTER and results are unpredictable if these requirements are not met.

6.4.1.3 General Coding Procedures

Several requirements must be followed when coding a format definition. The statement identifier, DISPLAY or FIELD, must be the first item coded. Parameters may be coded in any sequence within the statement and are separated from each other by a blank or comma. The parameter keyword and its value are separated by a blank or equals symbol. Extraneous blanks may appear before and after parameters, and surrounding separating equals symbols and commas.

Each statement will be coded onto punched cards or card images, using columns 1-71, inclusively. Care should be exercised when a statement keyword or value is coded through column 71. FORMATTER combines positions 1-71 from all records into a continuous area, and a required blank may need to be placed in column 1 of the next card. If a statement does not fit on one card, it may be continued onto any number of cards. A nonblank character in column 72 can be used to indicate a statement continuation, but it is not required.

If a parameter value includes a blank, comma, equals symbol, or single quote, it must be enclosed by single quotes. In addition, each single quote character included in the value must be represented by two consecutive single quote characters. This is necessary so that FORMATTER can distinguish delimiter quotes from quotes within the value. The pair of quotes will be reduced to a single quote when the value is processed.

6.4.2 Entering the Format Definition on the Library

A format definition may be entered into the user library in either the batch or on-line mode. The NIPS utility, UTODE, is used to enter the format definition in the batch mode. See NIPS Users Manual Volume VII, "Utility Support," for a discussion of this utility.

In the batch mode, the user must code the definition statements in punched cards or in card image records stored in a library. See NIPS Users Manual Volume VIII, "Job Preparation," for a sample job setup.

To add a format to the user library in the on-line mode, the definition source statements must first be added as a member to a library. This can be accomplished using the NIPS EDIT component. It should be noted that the name of the source statement member must be unique to the library and must not be the same as the name on the DISPLAY statement. The UPDATE command is used during a terminal session to add the format to the user library. This command is discussed in subsection 6.5.1.3.

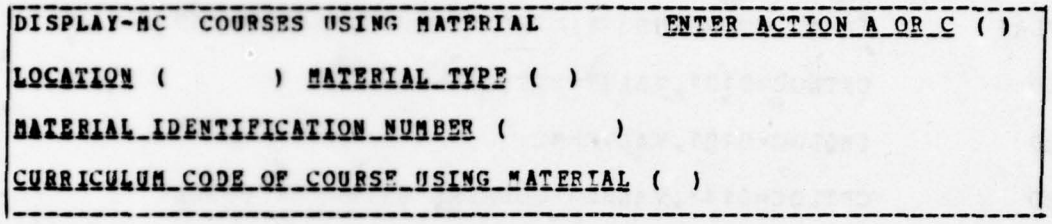
The user may add more than one format definition to the library at a time. This is done by simply placing one format definition after the other in the input source deck or in the source statement member.

6.4.3 Examples

Figure 15 shows a sample CRT display image and figure 16 shows the FORMATTER source statements used to define this format. This format is designed to invoke SODA after data entry, and uses the recycle option.

The IMQ record created from this format using operator-keyed data would be of the following form.

Record Position	Field Value
1-2	'MC'
3-3	'A' or 'C'
4-10	location
11-12	material type
13-19	material ID number
20-21	curriculum code



----indicates a high intensity display field
() indicates the position of a CRT/IMQ field and is not part of the format

Figure 15. Sample Display Format

```
DISPLAY     ID=MC,COMP=FMSODA,NEXT=EXECUTE,REPEAT=YES

FIELD       CRTLOC=0101,VALUE=DISPLAY-MC

FIELD       IMQLOC=0101,VALUE=MC

FIELD       CRTLOC=0113,VALUE='COURSES USING MATERIAL'

FIELD       CRTLOC=0144,VALUE='ENTER ACTION A OR C',
            HIGHLITE=YES

FIELD       CRTLOC=0165,CRTLEN=1,IMQLOC=0103,IMQLEN=1,
            DATA=REQUIRED

FIELD       CRTLOC=0301,VALUE=LOCATION,HIGHLITE=YES

FIELD       CRTLOC=0311,CRTLEN=7,IMQLOC=0104,DATA=REQUIRED

FIELD       CRTLOC=0320,VALUE='MATERIAL TYPE',HIGHLITE=YES

FIELD       CRTLOC=0335,CRTLEN=2,IMQLOC=0111,DATA=REQUIRED

FIELD       CRTLOC=0501,VALUE='MATERIAL IDENTIFICATION NUMBER',
            HIGHLITE=YES

FIELD       CRTLOC=0533,CRTLEN=7,IMQLOC=0113,DATA=REQUIRED,
            ATTRIBUTE=NUMERIC

FIELD       CRTLOC=0701,HIGHLITE=YES,
            VALUE='CURRICULUM CODE OF COURSE USING MATERIAL'

FIELD       CRTLOC=0743,CRTLEN=2,IMQLOC=0120,DATA=REQUIRED
```

Figure 16. Sample FORMATTER Definition

Figure 17 shows another sample CRT display image and figure 18 shows the FORMATTER source statements used to define this format. This format is an example of a format which contains lightpen selectable fields. After data entry, FMSODA is invoked.

The IMQ record created from this format using the operator-keyed data would be of the following form.

Record Position	Field Value
1-4	'FILE'
6-25	'NIPSM.TESTER' or 'NIPSM.PRIMARY'
30-35	'REPORT'
37-44	'TPTEST', 'REPT2' or 'TEST3A'

```
TERMINAL PROCESSING (TP)

     SELECT ONE FILE NAME FROM THE FOLLOWING LIST

          ?NIPSM.TESTER
          ?NIPSM.PRIMARY

     SELECT ONE REPORT FROM THE FOLLOWING LIST

          ?TPTEST        ?REPT2        ?TEST3A
```

Figure 17. Sample Format with Lightpen Selectable Fields

```
DISPLAY     ID=PEN1,COMP=FMSODA,NEXT=EXECUTE

FIELD       CRTLOC=0201,VALUE='SELECT ONE FILE NAME FROM
            THE FOLLOWING LIST'

FIELD       IMQLOC=0101,VALUE=FILE

FIELD       CRTLOC=0407,IMQLOC=0106,VALUE='NIPSM.TESTER',
            TYPE=L,IMQLEN=20

FIELD       CRTLOC=0507,IMQLOC=0106,VALUE='NIPSM.PRIMARY',
            CRTLEN=20,DATA=OPTIONAL,TYPE=LITEPEN,IMQLEN=20

FIELD       CRTLOC=0801,VALUE='SELECT ONE REPORT FROM THE
            FOLLOWING LIST'

FIELD       IMQLOC=0130,VALUE='REPORT'

FIELD       CRTLOC=1007,IMQLOC=0137,VALUE=TPTEST,TYPE=L,
            DATA=O,IMQLEN=8

FIELD       CRTLOC=1020,IMQLOC=0137,VALUE=REPT2,TYPE=LITPEN,
            DATA=O,IMQLEN=8

FIELD       CRTLOC=1033,IMQLOC=0137,VALUE=TEST3A,TYPE=L,
            DATA=O,IMQLEN=8
```

Figure 18. Sample FORMATTER Definition with Lightpen Selectable Fields

6.4.4 Dynamic Specification of Formats

The capability exists which allows the dynamic specification of the format sequence during a display session. After FORMATTER invokes QUIP or FMSODA, the QUIP query or the FMSODA logic statement may pass format names back to FORMATTER. These formats will be the next formats executed when FORMATTER recycles.

6.4.4.1 FMSODA Format Specification

The user's logic statement may pass from one to five format names back to FORMATTER. This is accomplished by placing the format names in designated positions in the FM workarea. Each time FORMATTER recycles, the first format name in the list is the next format which will appear on the CRT screen. This format name is then removed from the list and the list is 'pushed up.' Subsequent logic statements or QUIP queries may add format names to the end of the list. The format name list may contain a maximum of five names at any time.

This capability permits the user to determine his format sequence based upon data values in his file. For example, the user signs onto FORMATTER, specifies his initial format and enters data on the format. The format invokes FMSODA. The executed logic statement tests a field or fields and based on these tests stores a format name or names in the FM workarea. After FMSODA is finished, the first specified format is processed by FORMATTER.

To indicate to FMSODA that the FM workarea contains format names which are to be passed to FORMATTER, the user must store double dollar signs ($$) in the FM workarea locations W958/959. It is assumed that locations W958/959 will contain double dollar signs ($$) only in this circumstance. The format names must also be stored in specific locations of the FM workarea. These locations are as follows:

W958/959	'$$'
W960/967	format name 1
W968/975	format name 2

W976/983	format name 3
W984/991	format name 4
W992/999	format name 5

It is important to remember that the order of the format names determines the sequence in which they are processed by FORMATTER. Format name 1 is processed first and format name 5 is processed last.

The user is responsible for clearing the FM workarea locations W958/999. If format names have been stored by one logic statement execution and a subsequent execution does not clear these workarea locations or store the same number of format names, residual names from the earlier logic statement execution will be passed to FORMATTER. It is a wise practice to clear the workarea locations at the beginning of each logic statement.

To achieve the automatic display of the dynamically specified formats, each format, including the initial format, must be defined with REPEAT=YES.

6.4.4.2 FMSODA Parameter Specification

The user can also pass a parameter from his logic statement to FORMATTER. This is accomplished by storing the parameter and the length of the parameter in a specified location of the FM workarea. FMSODA transfers the parameter from the FM workarea to a 500-byte TP workarea. The user must also specify the desired position of the parameter in this TP workarea. To indicate to FMSODA that the logic statement has stored a parameter which is to be transferred to the TP workarea, the user must store double dollar signs ($$) in the FM workarea locations W800/801. It is assumed that double dollar signs ($$) will be placed in W800/801 only in this circumstance. The required locations in the FM workarea are as follows:

W800/801	'$$'
W802/804	Length of parameter
W805/807	Starting location in TP workarea
W808/957	Parameter

Note that the maximum length of a parameter is 150 characters.

By using the WORKLOC operator in a format definition, data in the TP workarea can be moved to the IMQ and/or to the CRT. The format can address any part of the 500-byte workarea.

The user thru his logic statement is completely responsible for the contents of the 500-byte TP workarea. Once data has been stored in the workarea, it will remain there until the user stores something else in the same location. Thus several formats may reference the same data in the TP workarea.

6.4.4.3 QUIP Format Specification

The user may pass from one to five format names from his QUIP query to FORMATTER. To accomplish this, the QUIP query or Load RIT must generate an output line that contains the characters, 'FORMAT=', in the first seven positions. The complete form of the line is as follows (with no intervening spaces):

$$\text{FORMAT} = \left[\begin{array}{l}\text{name} \\ (\text{name}^1, \text{name2} .. \text{name5})\end{array}\right]$$

The order in which the format names are provided, determines the sequence in which they are processed by FORMATTER. Format name1 is processed first and format name5 is processed last. If previous queries or logic statements have passed format names to FORMATTER, the names provided by the present query are stacked at the end of the list. The FORMATTER format name list may contain a maximum of five names at a time. As each format is processed, the name is removed from the top of the list.

The capability to dymamically specify formats in QUIP is most useful when FORMATTER recycles. Normally, because QUIP invokes PAGE after execution, FORMATTER would not recycle. By using two QUIP operands in his query, the user can

achieve automatic recycling when FORMATTER invokes QUIP and also save his query output for later use.

(1) VIEW = member name

This parameter causes the results of the query to be stored on the specified VIEW data set member. The member may be an existing member, in which case, the member is replaced or it may be a new member, in which case, the member is added. (The QUIP source statements will not appear in the VIEW member.)

(2) OMQ = YES / NO

This parameter is used to indicate whether the output from the QUIP query is to be placed on the OMQ. The default value is YES. If the user wishes to have the query output placed only on the VIEW data set, he must specify OMQ=NO.

These parameters must appear at the beginning of the query, before any QUIP operators. Note that the VIEW and OMQ parameters are not exclusive elements of dynamic format specification but may be used in any query.

6.4.4.4 Example

The following includes several examples demonstrating the use of dynamic format specification, the passing of a parameter from FMSODA to FORMATTER and the use of the QUIP parameters, VIEW and OMQ.

The following format invokes FMSODA with a transaction for logic statment ZZ.

```
DISPLAY     ID=FORMATA,NEXT=EXECUTE,COMP=FMOSDA,REPEAT=YES

FIELD       IMQLOC=0101,VALUE=ZZ

FIELD       CRTLOC=0205,VALUE='RECORD ID='

FIELD       CRTLOC=0217,CRTLEN=6,IMQLOC=0103,UNDERLINE=YES
```

```
FIELD          CRTLOC=0405,VALUE='RECORD TYPE='

FIELD          CRTLOC=0419,CRTLEN=3,IMQLOC=0109,UNDERLINE=YES
```

Note that the REPEAT=YES option is specified for this format. This is necessary because logic statement 'ZZ' will pass format names to FORMATTER and the user wishes to automatically recycle to the next format.

Suppose logic statement ZZ is as follows:

```
$ASP,TPTEST,ZZ,80
FIELD UIC 3  8  A  CONTROL  1
            POOL
            CLR  W800/999
            BNR  NEW
            MAL  T9/11,RECTYP    MOVE RECORD TYPE
            MAL  '$$',W958/959
            MAL  '$$',W800/801
            MAL  T3/8,W808/957    PARAMETER
            MAL  '005',W802/804   PARAMETER LENGTH
            MAL  '005',W805/807   STARTING LOCATION
            MAL  'UPDATEF',W960/967  FORMAT NAME1
            COA  T9/11,'XXX'       TEST RECORD TYPE
            BEQ  TYPEX
            MAL  'FORMATB',W968/975  FORMAT NAME2
            HLT
  TYPEX     MAL  'FORMATC',W968/975  FORMAT NAME3
            HLT
  NEW       DDR
            MAL  'FORMATD',W960/967  FORMAT NAME1
            MAL  '$$',W958/959
            HLT
            END
```

If the record ID is new, the format name 'FORMATD' is passed back to FORMATTER. FORMATD will automatically appear on the CRT screen. If the record ID is not new, the record ID is stored in the parameter area of the FM workarea and two format names are stored in the specified format name areas. The second format name is provided according to the transaction value of the record type. Note that W800/999 is cleared each time the logic statement is executed to prevent

residual data from previous executions being passed to FORMATTER.

Format UPDATEF is defined as follows:

```
DISPLAY         ID=UPDATEF,NEXT=EXECUTE,COMP=FMSODA,REPEAT=YES
FIELD           IMQLOC=0101,VALUE=UPDATE
FIELD           IMQLOC=0108,IMQLEN=6,WORLOC=5
```

This format uses the WORKLOC keyword to transfer the record ID passed by logic statement ZZ to the IMQ. FMSODA is invoked and the record is updated on the NIPS file. Note that format UPDATEF has no CRT fields defined. Thus FORMATTER will create the IMQ as defined, but no display will appear on the CRT screen. The next format the user will see on the screen will be either FORMATB or FORMATC.

FORMATB is defined as follows:

```
DISPLAY         ID=FORMATB,NEXT=EXECUTE,COMP=QUIP,REPEAT=YES
FIELD           CRTLOC=0205,VALUE='SUPPLY TWO FIELD NAMES FOR THE
QUIP QUERY.'

FIELD           CRTLOC=0409,CRTLEN=7,IMQLOC=0409,UNDERLINE=YES

FIELD           CRTLOC=0608,CRTLEN=7,IMQLOC=0414,UNDERLINE=YES

FIELD           IMQLOC=0101,VALUE='VIEW=REPORT1,OMQ=NO'

FIELD           IMQLOC=0201,VALUE='FILE TESTEP CLASS UNCLASSIFIED'

FIELD           IMQLOC=0301,VALUE='IF RECTYP NE ''XXX''.'

FIELD           IMQLOC=0401,VALUE=LIST

FIELD           IMQLOC=0501,VALUE='FINAL DISPLAY FORMAT=FORMATE'
```

This format invokes QUIP after data entry. The QUIP query output will be stored in the VIEW data set member REPORT1 and not on the OMQ. When the user wishes to see his output, he will invoke VIEW. The format, FORMATE will automatically appear on the CRT screen after QUIP completes.

6.4.5 Restrictions

There are two restrictions which should be remembered when using FORMATTER. First, all user library names must be unique within the library and must not be a FORMATTER reserve word. The FORMATTER reserve words are UPDATE, GETDATA and SIGNOFF. Second, the format name must be eight characters or less and must begin with a letter. Likewise, the format name must not be a FORMATTER reserve word.

6.5 Operational Phase

This subsection presents a description of a terminal session for the FORMATTER user and the terminal commands he may use during the session. An example is also given to clarify the operational phase functions.

6.5.1 Display Session

6.5.1.1 Initialization

After the CRT operator has begun a terminal session by logging into the NIPS TP system, he may invoke FORMATTER by entering ¬ (logical not symbol)O. When FORMATTER is given control, an initialization routine will be executed. The user will be asked to supply the user library name. The initialization routine will then present an introductory display on the CRT. See figure 19 for an example of this display. The introductory display will prompt the CRT operator to enter one of four possible commands: a format name, UPDATE, GETDATA or SIGNOFF. If the UPDATE command is given, a second display will follow. On this display the operator must supply the library and member names where his definition source statements are stored. See figure 20 for an example of this display. A second display will also follow the GETDATA command. The user must supply the data set name and if applicable, the member name of his external data source. Figure 21 shows this display. The user may also enter the command EXIT on the introductory command display. This command causes FORMATTER to enter a wait state and remain so until it is again invoked.

6.5.1.2 Format Name Command

The CRT operator will invoke a display session for data entry by keying in a format name. FORMATTER will recover and display the selected skeleton image. Then the operator may key data into the defined fields of the format. By adding the word TEST after the format name, the user can prevent the defined NIPS application program from being invoked. All other display session functions will be performed; thus, the user can debug his display image and IMQ creation by subsequently listing his IMQ.

COMMAND FORMAT (INTRODUCTION)

```
FORMATTER IS READY TO ACCEPT YOUR COMMAND.

THE ACCEPTABLE COMMANDS ARE. . .

    FORMAT-NAME - DISPLAY FORMATS, CREATE AN IMQ,
                  INVOKE A COMPONENT.

                - TO PREVENT NORMAL COMPONENT EXECUTION,
                  FOLLOW THE NAME WITH THE WORD "TEST".

         UPDATE - CREATE AND STORE NEW DEFINITIONS
                  IN THE LIBRARY.

        SIGNOFF - FREE ALL FORMATTER RESOURCES.

        GETDATA - READ AND STORE DATA INTO FORMATTER WORKAREA


ENTER YOUR COMMAND HERE. . .
```

Figure 19. Introductory CRT Display

```
                          UPDATE FORMAT


PLEASE KEY THE FULL DATA SET NAME OF YOUR SOURCE INPUT LIBRARY
AND THE NAME OF THE MEMBER THAT CONTAINS YOUR SOURCE DEFINITIONS.


DATA SET NAME. . .
MEMBER NAME. . .
```

Figure 20. UPDATE CRT Display

```
GETADATA FORMAT


PLEASE KEY THE FULL DATA SET NAME OF YOUR DATA INPUT SOURCE
AND THE NAME OF THE MEMBER.


DATA SET NAME ..


MEMBER NAME ..
```

Figure 21. GETDATA CRT Display

6.5.1.3 UPDATE Command

The UPDATE command permits the CRT operator to add format definitions to the user library in the on-line environment. The definition source statements must have been previously stored in a library (this can be accomplished with the use of TP EDIT). When entering the UPDATE command, the operator must also specify the name of the library member containing the definition source statements. FORMATTER will obtain the definition source statements, and write status and diagnostic messages on the OMQ. The PAGE component will then be invoked to allow the operator to review the OMQ.

6.5.1.4 GETDATA Command

The GETDATA command permits the FORMATTER user to read a data record from a data set and to move the data to the TP workarea. The purpose of this command is to give the user a means of communication between TP and any other independent online system.

The GETDATA command causes only the first record of the user specified data set to be read each time. The data set may be a sequential data set or a member of a partitioned data set. From one to give parameters pairs may be passed to FORMATTER. A parameter pair consists of a format name and a parameter value. FORMATTER expects the data record to be in the following form:

Record Position	Value
1-8	Format Name 1
9-10	Length of parameter 1
11-13	Offset in TP workarea to store parameter
14-14+n-1	Parameter value 1 (where n= length of parameter value 1)
14+n-14+n+7	Format Name 2
14+n+8-14+n+9	
.	.

. .
. .

The format names are moved to the FORMATTER format name list in the same order as they appear in the data record and the parameter values are moved to the specified TP workarea locations. By use of the FORMATTER parameter, WORKLOC, in the format definition, the parameter values may be moved to the IMQ and/or to the CRT.

After the execution of the GETDATA command, FORMATTER will automatically recycle to process the first format in the FORMATTER format name list.

6.5.1.5 SIGNOFF Command

When the operator foresees no further need for FORMATTER, he should enter the SIGNOFF command so that the allocated resources may be released.

6.5.1.6 Display Action Code

The display action code is specified by the NEXT parameter in the definition phase. This code will appear on all displays in a reserved area (line 24, positions 71-80). The CRT operator may override the defined code by keying into this area a format name, or the word EXECUTE, SIGNOFF or EXIT. If the word EXECUTE is specified, the designated NIPS application will be invoked. The word EXIT will cause the display session to be cancelled and all IMQ data created up to this point to be scratched. SIGNOFF will cause all FORMATTER resources to be released.

6.5.2 Error Procedures

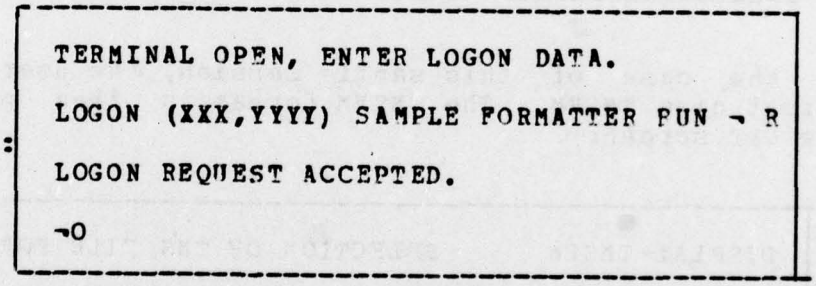
FORMATTER will execute an error procedure if required data is not keyed, or if a user subroutine or table returns a code indicating unsuccessful processing. If a field was defined as REQUIRED in the DATA parameter of the FIELD statement, the operator must key in data. If he does not, the display will be rewritten on the CRT with all the keyed

data and with asterisks in the required field. The CRT operator must then key into this field; failure to do so will result in a repeat of the error procedure. In the case of a subroutine error, the subsequent action depends upon the value of the DATA parameter. If the field data is required, the same procedure just described will occur. If the field data is optional, FORMATTER will store blanks in the IMQ field.

6.5.3 Example of Terminal Session

This subsection presents an example of a terminal session. A sample of the CRT screen is shown in each step of the session.

The user must first log onto the TP system. After successfully logging on, the user will invoke FORMATTER.



The user invoked FORMATTER by entering ¬ (logical not)O. FORMATTER then presents an introductory display on the CRT.

```
FORMATTER IS READY TO ACCEPT YOUR COMMAND.

THE ACCEPTABLE COMMANDS ARE. . .

   FORMAT-NAME - DISPLAY FORMATS, CREATE AN IMQ,
                 INVOKE A COMPONENT.

               - TO PREVENT NORMAL COMPONENT EXECUTION,
                 FOLLOW THE NAME WITH THE WORD "TEST."

   UPDATE      - CREATE AND STORE NEW DEFINITIONS
                 IN THE LIBRARY.

   SIGNOFF     - FREE ALL FORMATTER RESOURCES.

ENTER YOUR COMMAND HERE. . . TMSFM
```

In the case of this sample session, the user entered the format name TMSFM. The TMSFM format is then presented on the CRT screen:

```
DISPLAY-TMSFM      SELECTION OF TMS FILE FOR MAINTENANCE

FROM THE FOLLOWING LIST SELECT THE TMS FILE TO BE UPDATED.
ALL DISPLAYS REQUESTED FROM THIS POINT UNTIL AN UPDATE
COMMAND HAS BEEN ISSUED MUST PERTAIN TO THE FILE THAT
IS ENTERED.  THE LEGITIMATE ENTRIES ARE - ASSETS, CLASSP,
COURSE, KEY, MISSED, PEOPLE, QDB, TNGMAT, TPO.

FILE SELECTED IS (      )


                                                  EXECUTE
```

Following the directions of the TMSFM format, the CRT operator keys in the file name, TNGMAT. An IMQ record is created according to the specifications of the format definition.

```
DISPLAY-TMSFM        SELECTION OF TMS FILE FOR MAINTENANCE

FROM THE FOLLOWING LIST  SELECT THE TMS FILE TO BE UPDATED.
ALL DISPLAYS REQUESTED FROM THIS POINT UNTIL AN UPDATE
COMMAND HAS BEEN ISSUED MUST PERTAIN TO THE FILE THAT
IS ENTERED.  THE LEGITIMATE ENTRIES ARE - ASSETS, CLASSP,
COURSE, KEY, MISSED, PEOPLE, QDB, TNGMAT, TPO.

FILE SELECTED IS (TNGMAT )



                                                    EXECUTE
```

After data entry on the TMSFM format, FORMATTER automatically invokes FMSODA. Note that the display action code on the CRT image specifies EXECUTE. In the format definition of TMSFM, the value FMSODA was coded for the COMP parameter. After FMSODA executes, a message is written on the CRT and FMSODA enters a wait state.

```
Response:  EOM RECEIVED.
           FMSODA STARTED.
           ---TNGMAT---REPORTID---
           INPUT SCRATCHED. READY FOR MORE.
Enter:     ¬O
```

The user, wishing to invoke FORMATTEP, again keys in ¬ (logical not) O. Each time FORMATTER is invoked by the CRT operator, the introductory display is presented. The user now wishes to use format MC to enter data. He enters his command on the introductory display.

```
FORMATTER IS READY TO ACCEPT YOUR COMMAND.

THE ACCEPTABLE COMMANDS ARE. . .

 FORMAT-NAME - DISPLAY FORMATS, CREATE AN IMQ,
               INVOKE A COMPONENT.

             - TO PREVENT NORMAL COMPONENT EXECUTION,
               FOLLOWING THE NAME WITH THE WORD "TEST."

  UPDATE     - CREATE AND STORE NEW DEFINITIONS
               IN THE LIBRARY.

  SIGNOFF    - FREE ALL FORMATTER RESOURCES.

ENTER YOUR COMMAND HERE. . . MC
```

The MC format is then presented on the CRT screen. (See figure 16, section 6.4.3 for the definition source statements of format MC.)

```
DISPLAY-MC COURSES USING MATERIAL    ENTER ACTION A OR C ( )
 LOCATION (       ) MATERIAL TYPE ( )
MATERIAL IDENTIFICATION NUMBER (       )
  CURRICULUM CODE OF COURSE USING MATERIAL (   )
                                                  EXECUTE
```

The user enters his data on the format in the specified locations. FORMATTER creates an IMQ record according to the specifications of the MC format definition.

```
 DISPLAY-MC COURSES USING MATERIAL   ENTER ACTION A OR C (A)
 LOCATION (XXXXXXXX) MATERIAL TYPE (XX)
MATERIAL IDENTIFICATION NUMBER (9999999)
 CURRICULUM CODE OF COURSE USING MATERIAL (XX)
                                                  EXECUTE
```

FORMATTER now automatically invokes SODA. SODA processes the IMQ record. At this point, because the MC format definition specified the recycle option, the MC format is again presented on the CRT without any user action. The user has finished entering his data and no longer needs the MC format. Thus he overrides the display action code by keying in EXIT in the reserved area of line 24.

```
DISPLAY-MC COURSE USING MATERIAL   ENTER ACTION A OR C ( )

LOCATION (     ) MATERIAL TYPE ( )

MATERIAL IDENTIFICATION NUMBER (        )

CURRICULUM CODE OF COURSE USING MATERIAL (   )

                                                        EXIT
```

FORMATTER will now bring the introductory command format up on the CRT screen. The user does not wish to use FORMATTER any longer. To signoff of FORMATTER, he enters the word, SIGNOFF, on the introductory display.

```
 FORMATTER IS READY TO ACCEPT YOUR COMMAND.

 THE ACCEPTABLE COMMANDS ARE. . .

FORMAT-NAME - DISPLAY FORMATS, CREATE AN IMQ,
              INVOKE A COMPONENT.

            - TO PREVENT NORMAL COMPONENT EXECUTION,
              FOLLOW THE NAME WITH THE WOPD "TEST."

UPDATE      - CREATE AND STORE NEW DEFINITIONS
              IN THE LIBRARY.

SIGNOFF     - FREE ALL FORMATTER RESOURCES.


 ENTER YOUR COMMAND HERE. . . SIGNOFF
```

The FORMATTER resources have now been released. The updated record processed by SODA from the data entered by FORMATTER is on a temporary hold file. Since the user has entered all of his data, he now wishes to merge his update records into the TNGMAT file.

Enter:

```
UPDATE ALL ¬F
```

Response:

```
EOM RECEIVED.
FMSODA STARTED.

ALL UPDATED RECORDS WRITTEN ON TNGMAT.
SIGNOFF ACCEPTED TNGMAT.
```

The user has now signed off of SODA and has finished the work that he wanted to accomplish during this terminal session. He may now logoff of the TP system:

```
Enter:
                LOGOFF ¬R
Response:
                LOGOFF REQUEST ACCEPTED.
```

The terminal session is now completed.

6.6 System Flow Summary

Figures 22 and 23 present a system overview of the definition and operational phases of FORMATTER. The schematics show the I/O flow during each phase.

TERMINAL PROCESSING (TP)

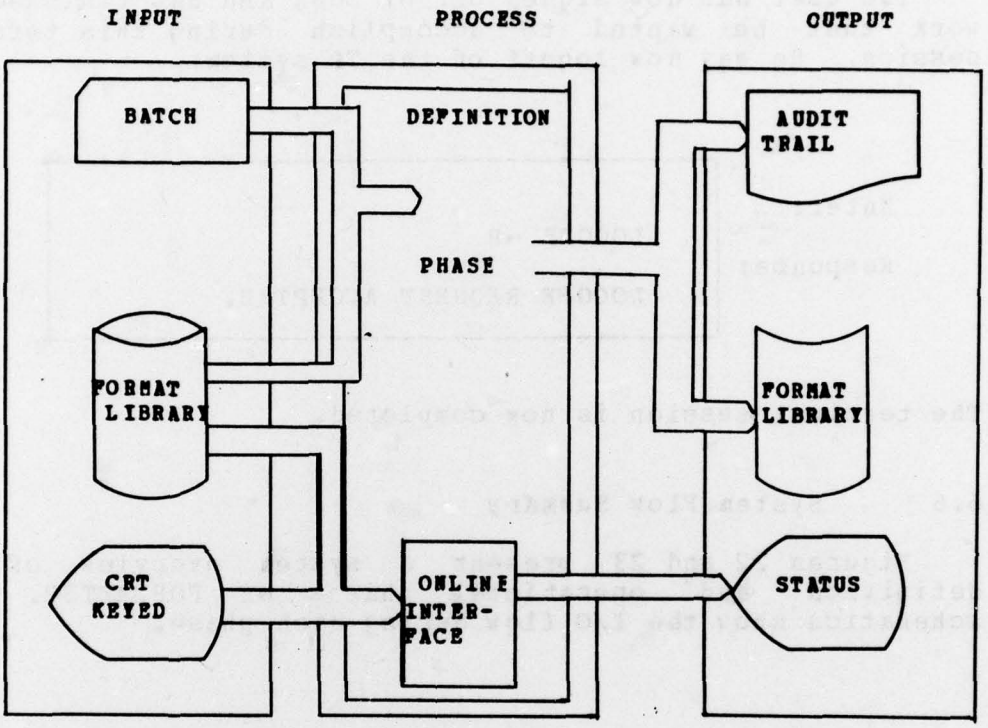


Figure 22. Operational Phase Overview

TERMINAL PROCESSING (TP)

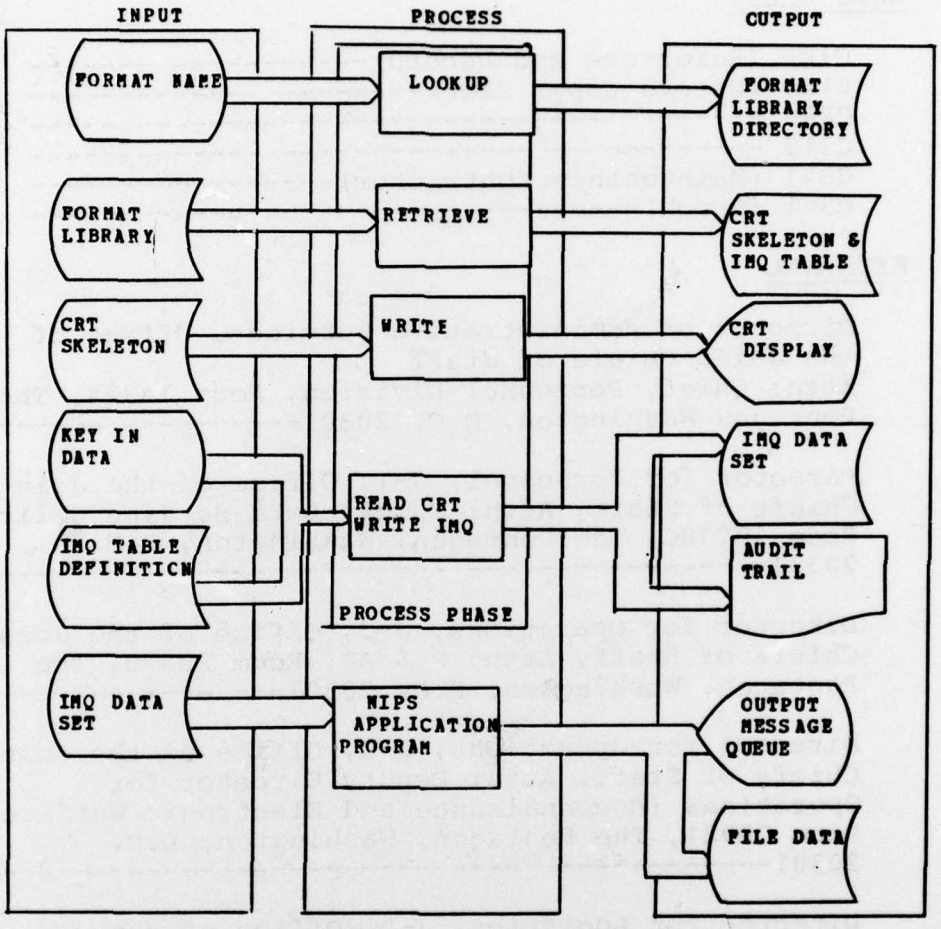


Figure 23. Operational Phase Overview

DISTRIBUTION

CCTC CODES	COPIES
C124 (Reference and Record)	3
C124 (Record Copy) Stock	6
C240	20
C315	1
C341 (Maintenance Contractor)	10
C341 (Stock)	70

EXTERNAL

Director of Administrative Services, Office of the Joint Chiefs of Staff
Attn: Chief, Personnel Division, Room 1A724, The Pentagon Washington, D.C. 20301---------- 1

Director for Personnel, J-1, Office of the Joint Chiefs of Staff, Attn: Chief, Data Service Office, Room 1B738C, The Pentagon, Washington, D.C. 20301---------- 1

Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: P & AD, Room 2B870, The Pentagon, Washington, D.C. 20301---------- 1

Director for Operations, J-3, Office of the Joint Chiefs of Staff, Attn: Deputy Director for Operations (Reconnaisance and Electronic Warfare) Room 2D921, The Pentagon, Washington, D.C. 20301---------- 1

Director for Logistics, J-4, Office of the Joint Chiefs of Staff, Room 2E828, The Pentagon, Washington, D.C. 20301---------- 1

Chief, Studies Analysis and Gaming Agency, Attn: Chief, Force Analysis Branch, Room 1D928A, The Pentagon, Washington, D.C. 20301---------- 1

Automatic Data Processing, Liaison Office National Military Command Center, Room 2D901A, The Pentagon, Washington, D.C. 20301---------- 1

EXTERNAL	COPIES
Automatic Data Processing Division Supreme Headquarters Allied Powers, Europe Attn: SA & P Branch, APO New York 09055	1
Director, Defense Communications Agency, Office Of MEECN System Engineering, Attn: Code 960T, Washington, D.C. 20301	1
Director, Defense Communications Engineering Center, Hybrid Simulation Facility, 1860 Wiehl Avenue, Reston, VA 22070	1
Director, Defense Intelligence Agency Attn: DS - 5C2 Washington, D.C. 20301	5
Commander-in-Chief, Pacific, Attn: J6331, FPO San Francisco, 96610	1
Commander-in-Chief, US Army Europe and Seventh Army ATTN: OPS APO New York 09403	1
Commanding General, US Army Forces Command, Attn: Data Support Division, Building 206, Fort McPherson, GA 30303	1
Commander, Fleet Intelligence Center, Europe, Box 18, Naval Air Station, Jacksonville, Florida 32212	1
Commanding Officer, Naval Air Engineering Center, Ground Support Equipment Department, SE 314, Building 76-1, Philadelphia, PA 19112	1
Commanding Officer, Naval Security Group Command, 3801 Nebraska Avenue, N.W. Attn: GP22, Washington, D.C. 20390	1
Commanding Officer, Navy Ships Parts Control Center, Attn: Code 712, Mechanicsburg, PA 17055	1
Headquarters, US Marine Corps, Attn: System Design and Programming Section (MC-JSMD-7) Washington, D.C. 20380	1

EXTERNAL	COPIES
Commanding Officer, US Army Forces Command Intelligence Center, Attn: AFIC-PD, Fort Bragg, NC 28307----------	1
Commander, US Army Foreign Science and Technology Center, Attn: AMXSJ-CS, 220 Seventh Street NE, Charlottsville, VA 22212--	1
Commanding Officer, US Army Security Agency, Command Data Systems Activity (CDSA) Arlington Hall Station, Arlington, VA 22212----------	1
Commanding Officer, US Army Security Agency Field Station - Augsburg, Attn: IAEADP, APO New York 09458----------	1
Commander, Fleet Intelligence Center, Atlantic, Attn: DPS, Norfolk, VA 23511----------	1
Commander, Fleet Intelligence Center, Pacific, Box 500, Pearl Harbor, HI 96860----------	1
Air Force Operations Center, Attn: Systems Division (XOOCSC) Washington, D.C. 20301-----	1
Commander, Armed Forces Air Intelligence Training Center, TTMNIM (360 FFS), Lowry AFB, Co 80230----------	1
Commander, Air Force Data Services Center, Attn: Director of System Support, Washington, D.C. 20330----------	1
Commander-in-Chief, US Air Forces in Europe, Attn: ACDI APO New York 09332----------	1
Commander, USAF Tactical Air Command, Langley AFB, VA 23665----------	1
Commander, Space and Missile Test Center, Attn: (ROCA) Building 7000, Vandenberg, AFB, CA 93437----------	1

EXTERNAL	COPIES
Naval Air Systems Command, Naval Air Station, Code 13999, Jacksonville, Florida 32212------	1
Commanding General, US Army Computer Systems Command, Attn: Support Operations Directorate, Fort Belvoir, VA----------	1
Defense Documentation Center, Cameron Station, Alexandria, VA 22314----------	12
TOTAL	159

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CSM UM 15-78, Volume VI	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NMCS Information Processing System 360 Formatted File System (NIPS 360 FFS) - Users Manual Vol VI - Terminal Processing (TP)		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s) DCA 100-77-C-0065
9. PERFORMING ORGANIZATION NAME AND ADDRESS International Business Machines, Corp. Rosslyn, Virginia		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS National Military Command System Support Center The Pentagon, Washington, D.C. 20301		12. REPORT DATE 1 September 1978 13. NUMBER OF PAGES 349
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Copies of this document may be obtained from the Defense Documentation Center, Cameron Station, Alexandria, Virginia 22314.
This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This volume defines the Terminal Processing (TP) component of the NIPS 360 FFS. It describes the on-line capabilities and language specifications for effective use of the component. The TP Monitor/Supervisor and specific TP Application Programs are discussed.

This document supersedes CSM UM 15-74, Volume VI (TP)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)