

AD-A056 207

NATIONAL MILITARY COMMAND SYSTEM SUPPORT CENTER WASH--ETC F/G 9/2  
NMCS INFORMATION PROCESSING SYSTEM 360 FORMATTED FILE SYSTEM (N--ETC(U)  
JUN 75

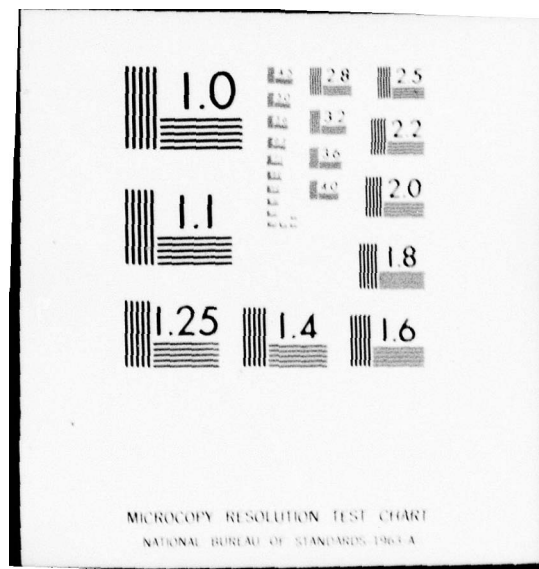
UNCLASSIFIED

NMCSSC-CSM-UM-15-74-V3-1/

NL

1 OF 1  
AD  
A056 207







DEFENSE COMMUNICATIONS AGENCY  
NATIONAL MILITARY COMMAND SYSTEM  
SUPPORT CENTER  
WASHINGTON, D. C. 20301

6 NMCS Information Processing System  
360 Formatted File System (NIPS 360 FFS)  
- Users Manual. Volume III. File  
Maintenance (FM). Changes 1, and 2.

IN REPLY  
REFER TO:

TO: DISTRIBUTION

SUBJECT: Change 1 to CSM UM 15-74, File Maintenance  
Volume III, dated 15 October 1974

14 NMCS-SSC-CSM-UM-15-74-V3-1/2

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.
2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.
3. When this change has been posted, make an entry in the Record of Changes on the inside cover.

FOR THE COMMANDER:

13 Enclosures  
Change 1 pages

*J. Douglas Potter*  
J. DOUGLAS POTTER  
Chief, Military Personnel  
and Administrative  
Services Office

DISTRIBUTION STATEMENT A

Approved for public release  
Distribution Unlimited

DDC  
RECEIVED  
JUL 14 1978  
RECEIVED

① 2B

243 000

JB

AD A056207

AD No.             
DDC FILE COPY

LEVEL III

12 53p.

EFFECTIVE PAGES - 1 June 1975

This list is used to verify the accuracy of CSM UM 15-74, Volume III, after change pages have been inserted. Original pages are indicated by the letter O, change 1 by the numeral 1.

<u>Page No.</u>	<u>Change No.</u>
TITLE	O
ii - iii	O
iv - v	1
vi	O
vii	1
viii - 26	O
27 - 30	1
31 - 132	O
133 - 134	1
134.1 - 134.2	1
135 - 166	O
167 - 168	1
169 - 200	O

EX-15-74-15	
ATB	ATB Section <input checked="" type="checkbox"/>
AMS	AMS Section <input type="checkbox"/>
UNP/AMH/MSD	<input type="checkbox"/>
DISTRIBUTION	
BY	
DISTRIBUTION AVAILABILITY CODE	
Dist.	AVAIL. and/or SPECIAL
A	

③ 48



Section		Page
4.2	Transactions.....	23
4.3	Subroutine Library.....	25
4.4	Data File.....	25
5	OUTPUTS.....	26
5.1	Output Data File.....	26
5.2	Auxiliary Output.....	26
5.2.1	Tape and Disk Auxiliary Output.....	26
5.2.2	Punched Card Auxiliary Output.....	26
5.2.3	Printed Auxiliary Output.....	27
5.3	Run History.....	27
5.4	File Analysis and Run Optimization Statistics.....	27
6	CONTROL CARD FORMATS.....	30
6.1	Free-Format Specifications.....	30
6.1.1	FMS Control Card (Free-Format).....	31
6.1.2	Segment Control Cards.....	34
6.1.3	Library Action Card (Free-Format)...	35
6.1.4	Transaction Descriptor (TD) Cards (Free-Format).....	38
6.1.5	Language Identifier Card.....	40
6.1.6	Logic Statement END Card.....	40
6.1.7	Logic Statement Library Update Terminator Card.....	40
6.1.8	Report Identifier Card.....	40
6.2	Ordinary Maintenance (CM) Transaction Descriptor (TD) Cards...	41
6.2.1	Keyword: FIELD.....	42
6.2.2	Keyword: CONTROL.....	43
6.2.3	Keyword: PICTURE.....	44
6.2.4	Keyword: VALUE.....	44
6.2.5	Keyword: RANGE.....	45
6.2.6	Keyword: VERIFY.....	46
6.2.7	Keyword: CONVERT.....	46
6.2.8	Keyword: GENERATE.....	47
6.2.9	Keyword: ERROR.....	48
6.3	Fixed-Format Specifications.....	50
6.3.1	FMS Control Card.....	50
6.3.2	Library Action Cards.....	52
6.3.3	Transaction Descriptor (TD) Cards.....	54
6.3.4	Language Identifier Card.....	56
7	ICCL LANGUAGE.....	58
7.1	Card Format.....	58
7.1.1	Symbols.....	58
7.1.2	Operators.....	58

## Section

## Page

7.1.3	Operands.....	58
7.1.4	Comments.....	58
7.2	Operand Coding.....	59
7.3	FOOL Instructions.....	61
7.3.1	Alphabetical Listing.....	61
7.3.2	Valid Operands Chart.....	68
7.3.3	Instruction Groups.....	73
7.4	ECCL Instructions.....	80
7.4.1	Environment Handling Instructions...	80
7.4.2	Data Handling Instructions.....	83
7.4.3	Control Instructions.....	92
7.4.4	Display Instructions.....	100
7.4.5	Ordinary Maintenance Validity Test Instructions.....	102
7.4.6	Transaction Error Log Instruction (SODA and CM).....	103
7.5	Logic Statement Examples.....	104
7.5.1	FMS Control Card.....	104
7.5.2	Library Action Card to Add a Report..	104
7.5.3	Logic Statement Setup.....	105
7.5.4	Use of Data Conversion Subroutines...	109
7.5.5	Periodic Set Processing.....	111
7.5.6	Test for Numeric Data.....	115
7.5.7	Production of Summary Information....	118
7.5.8	Variable Field and Set Processing....	121
7.6	Summary of ECCL Instructions.....	125
8	ORDINARY MAINTENANCE (CM) EXAMPLES...	128
8.1	Use of Ordinary Maintenance TI Cards.	128
8.2	Use of Ordinary Maintenance TI Cards and ECCL Instructions.....	128
9	NEW FILE MAINTENANCE LANGUAGE (NFL) .	130
9.1	NFL Statement Composition.....	130
9.1.1	Statement Identifiers.....	131
9.1.2	Keywords.....	132
9.1.3	Noise Words.....	134
9.1.4	Statement Labels.....	134
9.1.5	Operands.....	134.1
9.1.5.1	Control Location Operands.....	135
9.1.5.2	Subroutine/Table Name Operands.....	135
9.1.5.3	Literal Value Operands.....	135
9.1.5.4	Data Location Operands.....	136
9.1.5.4.1	File Data Operands.....	137
9.1.5.4.2	Transaction Data Operands.....	137
9.1.5.4.3	Indirect Data Operands.....	137
9.1.5.4.4	Defined Constant and Area Operands.	138

Section		Page
9.3.3	Control Point Identifiers.....	166
9.3.3.1	The NOTE Statement.....	166
9.3.3.2	The PROCEDURE Statement.....	167
9.3.3.3	The ENI Statement.....	167
9.3.3.4	The ELSE Statement.....	167
# 9.3.3.5	The CONTINUE Statement.....	168
9.3.3.6	The Language Identifier Statement...	168
9.4	NFL Logic Statement Examples.....	169
9.4.1	FMS Control Card.....	169
9.4.2	Library Action Card to Add a Report.	170
9.4.3	Logic Statement Setup.....	170
9.4.4	Use of Data Conversion.....	175
9.4.5	Periodic Set Processing.....	177
9.4.6	Test for Numeric Data.....	180
9.4.7	Production of Summary Information...	182
9.4.8	Variable Field and Variable Set Processing.....	184
9.5	Summary of NFI Condition and Action Statement Syntax.....	186
APPENDIX	Utilizing a NIPS File as FM Transaction Input.....	193
	DISTRIBUTION.....	195



## FILE MAINTENANCE (FM)

### 5.2.3 Printed Auxiliary Output

The FM component provides the user with two printed outputs. The user is responsible for formatting the print lines. The FM component will handle the printing of 132 characters or less. If the user requests that more than 132 characters of data be printed, the data in excess of 132 characters will be printed on subsequent lines.

### 5.3 Run History

As part of its functioning, the FM component automatically generates a run history on the printer. This includes listings of logic statements that are compiled, and messages indicating that errors, or unusual conditions that might be interpreted as errors, have been encountered during processing.

If segmented file processing is being performed, the segment control records on the current segment will be printed showing the segment boundary and the volume serial number of each segment.

### 5.4 File Analysis and Run Optimization Statistics

# The File Analysis Statistics capability in the FM component provides transactions showing the number of times each logic statement is executed during an FM execution. The DSNNAME of this data set must be the data file name plus a T. The T is added to ISAM names; the S is replaced by T in SAM names. To obtain transaction output, the DSNNAME 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 data set exists at execution time, transactions will be added (DISP=MOD). If it does not exist, five tracks will be dynamically allocated. The user may change this 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 DSNNAME (TRANS) in the TRANS DD card, he must supply all parameters required to process the

⑨

## FILE MAINTENANCE (FM)

data set. These parameters must conform to the requirements defined here.

The Run Optimization Statistics capability provides the user with statistical data reflecting the core allocation during FM execution. The breakdown of the statistics detail the amount of core used for user subroutines and tables, logic statements, process blocks, I/O buffers, and access methods. It also includes the number of FLDL entries allocated and used and the number of entries required for each subroutine, table, and logic statement to reside in core. The amount of core required for each subroutine, table, and logic statement to reside in core is also output. If subroutines, tables, and logic statements are rolled, this information will be output with the causes for the rolling and the number of times it occurred.

In addition, the user is able to enter override parameters for the number of BIDL entries to allocate, and the size of the processing block desired for storage of the data records during FM processing.

The statistics gathering is initiated through parameters entered in the PARAM field of the EXEC card. The parameters and their functions are as follows:

ROS	-	Indicates that run optimization information is to be gathered and output.
NOROS	-	Indicates that optimization processing is to be omitted. If no other parameters are coded, this parameter should be omitted as it is the default.

The parameters the user may supply to tailor his core allocation are listed below. Using these parameters, the Run Optimization Statistics are gathered and output unless the NOROS parameter is used.



## FILE MAINTENANCE (FM)

TCP=NK	-	The N indicates the number of bytes requested for the process block in 1000 (K) bytes.
TCB=n	-	The n indicates the number of entries to be used in the ELCL list for SUBSTP, the subroutine supervisor.
TCS	-	This parameter indicates that the statistics record on the ISAM data file is to be used to compute the process block size. This parameter must not be used with TCP and vice versa.

For a more detailed description of the capability see Introduction to File Concepts, Volume 1.

## FILE MAINTENANCE (FM)

### Section 6

#### CONTROL CARD FORMATS

##### 6.1 Free-Format Specifications

This section specifies the preparation requirements for all FM control cards. These cards may be punched in free format or fixed format (see sections 6.1 and 6.3 respectively).

The general rules that apply to free-format control cards are as follows:

- a. The control card data must always be punched starting in column 1. The first character of a control card must always be a dollar sign (\$).
- b. The information in the cards must be punched in a specified parameter sequence.
- c. The control card fields must be separated by commas, with no intervening blanks.
- d. If the analyst has no requirement for a certain parameter, he must so indicate by punching a comma for that field, except when he has no more fields to punch.

The three control cards that may be formatted in this manner are as follows:

- a. The FMS control card
- b. The Library Action card
- c. The Transaction Descriptor card.

## FILE MAINTENANCE (FM)

NEW RECORD Identifies a new record condition

JOB COMPLETE Identifies a run complete condition  
JOB COMPLETED

OVERFLOW Identifies an overflow condition

<u>Keyword</u>	<u>Usage</u>
----------------	--------------

+	Add operation
---	---------------

-	Subtract operation
---	--------------------

/	Divide operation
---	------------------

*	Multiply operation
---	--------------------

=	Denotes equivalency in compute statement
---	--

ON (following PRINT, PUNCH, WRITE, DISPLAY) signifies that a device is being specified

EXIT Identifies the next operand as an exit label

RECORD Identifies the object of the action as a record

FIELD Identifies the object of the action as field

SET Identifies the object of the action as a set

SUBSET Identifies the object of the action as a subset

OVER Specifies resulting position of set.

FAST

BEYOND

NEXT

FIRST

## FILE MAINTENANCE (FM)

### 9.1.3 Noise Words

Noise words are a group of commonly used words which have no effect on the functions of the statement but which help to give the statements a more English-like readability. Noise words can appear any place within a statement. The following noise words are allowed in NFI:

A	FOR	IS	WITH
AN	FROM	THEN	USING
AS	IN	THE	
BY	INTO	TO	

### 9.1.4 Statement Labels

The provision for labeling statements (or procedures) allows a name to be associated with a statement. These names can then be referenced as exit points or to change the sequence of execution. Labels must begin with an alpha character and may contain only alphanumeric characters. The label can contain up to seven characters. It is identified as a label by suffixing the name with a colon. A valid label must have a space following the colon. Statement identifiers or noise words cannot be used as labels.

Example -

```
LOOP: MOVE....GC TO LCOF
```

LOOP is a label associated with the MOVE statement. In the example, the GC causes the order of execution to be changed to the statement labeled LCOF.

\* The following statements may not have labels:

An IF statement or any part of an IF statement such as the AND, OR, OR ELSE clause.

A CONTINUE statement

A NCTE statement

A DEFINE statement

14



## FILE MAINTENANCE (FM)

### 9.1.5 Operands

Statement operands identify control locations, subroutines and tables, literal values, and data locations.



## FILE MAINTENANCE (FM)

### 9.3.3.2 The PROCEDURE Statement

The PROCEDURE statement identifies the beginning of a group of statements which will be treated as a unit. The PROCEDURE must be preceded by a label. This label is called the procedure name.

### 9.3.3.3 The END Statement

The END statement identifies two control points. It identifies the end of a procedure and/or it identifies the end of the logic statement. It has no operands.

Example -

NFI Conditions and Actions

```
PROC1: PROCEDURE  
Conditions and Actions  
END
```

```
PROC2: PROCEDURE  
Condition and Actions  
END
```

END.

The preceding example illustrates a logic statement containing two procedures. The first END statement encountered terminates the group of statements which comprise procedure PROC1. The second END statement does the same for the procedure PROC2. The third END statement terminates the entire logic statement.

### 9.3.3.4 The ELSE Statement

The ELSE statement identifies the beginning of a group of action statements which are to be executed only if a preceding condition was false. This statement is not required and it should only be used when there are both true and false actions (true actions consist of those actions immediately following the condition and continuing until an ELSE or CONTINUE statement is encountered). False actions

(17)

## FILE MAINTENANCE (F1)

commence with the first action following the ELSE statement and continue until a CONTINUE statement is encountered.

Example -

```
...IF THE CCUNT IS GREATER THAN 10 COMPUTE CCUNT = CCUNT  
*2 ELSE MOVE ZEROS TO CCUNT CONTINUE...
```

In the preceding example, if the condition is true, the COMPUTE statement would be executed. If it is false, the MOVE statement would be executed. Any number of actions could appear where these two actions appear.

### 9.3.3.5 The CONTINUE Statement

# The CONTINUE statement identifies the point, following a condition, that execution of nonconditional statements is resumed. There are no exceptions; each IF statement must have associated with it a corresponding CONTINUE statement. All NFL statements which follow an IF statement are considered to be part of the IF statement. To terminate an IF block, a CONTINUE statement is required.

Example -

```
...IF FIELD IS NOT EQUAL TO BLANKS MOVE FIELD TO REC  
CONTINUE...
```

In the example above, if the condition is true, execution is resumed with the next statement following the CONTINUE after the true actions are performed. If the condition is false, execution is resumed with the next statement following the CONTINUE statement.

Example -

```
...IF FIELD IS NOT EQUAL TO BLANKS MOVE FIELD TO REC  
ELSE MOVE REC TO FIELD CONTINUE...
```

In the preceding example, if the condition is true, processing is the same as in the previous example. However, if the condition is false, the false actions (those between the ELSE statement and the CONTINUE statement) are executed; execution then is resumed with the next statement following the CONTINUE statement.



**DEFENSE COMMUNICATIONS AGENCY**

COMMAND AND CONTROL  
TECHNICAL CENTER  
WASHINGTON, D. C. 20301

10 June 1976

IN REPLY  
REFER TO:

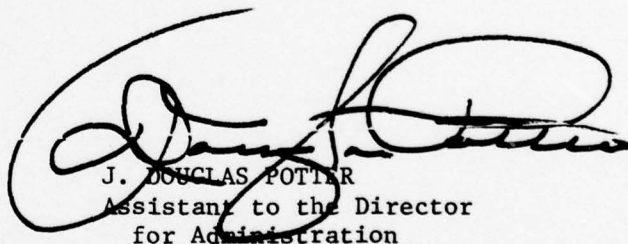
TO: DISTRIBUTION

SUBJECT: Change 2 to CSM UM 15-74, File Maintenance (FM),  
Volume III, dated 15 October 1974

1. Insert the enclosed change pages and destroy the replaced pages according to applicable security regulations.
2. A list of Effective Pages to verify the accuracy of this manual is enclosed. This list should be inserted before the title page.
3. When this change has been posted, make an entry in the Record of Changes on the inside cover.

FOR THE DIRECTOR:

40 Enclosures  
Change 2 pages

  
J. DOUGLAS POTTER  
Assistant to the Director  
for Administration



EFFECTIVE PAGES - 10 June 1976

This list is used to verify the accuracy of CSM UM 15-74, Volume III, after change 2 pages have been inserted. Original pages are indicated by the letter O, change 2 by the numeral 2.

<u>Page No.</u>	<u>Change No.</u>
Title Page	2
ii	O
iii - vii	2
viii	O
1 - 2	O
3 - 8	2
9 - 10	O
11 - 20	2
20.1 - 20.2	2
21 - 26	O
27 - 30	2
31 - 32	O
33 - 34	2
34.1 - 34.2	2
35 - 42	O
43 - 44	2
45 - 102	O
103 - 106	2
106.1 - 106.2	2



COMMAND AND CONTROL TECHNICAL CENTER  
Computer System Manual Number CSM UM 15-74

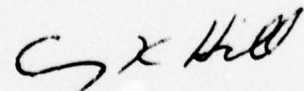
10 June 1976

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

Users Manual

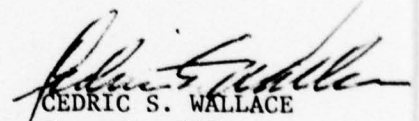
Volume III - File Maintenance (FM)

SUBMITTED BY:



CRAIG K. HILL  
Captain USA  
CCTC Project Officer

APPROVED BY:



CEDRIC S. WALLACE  
Captain, USN  
Deputy Director, NMCS ADP

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

CH-2



# CONTENTS

Section		Page
	ACKNOWLEDGMENT.....	ii
	ABSTRACT.....	viii
1	INTRODUCTION.....	1
# 2	FM CAPABILITIES.....	2
2.1	Transaction Sources.....	3
2.2	Transaction Formats.....	3
2.3	Logical Updating and Trans- action Editing.....	3
2.4	Data Conversion and Validation - User Subroutines.....	4
2.5	Processing of Periodic Sets.....	4
2.6	Variable Field and Variable Set Maintenance.....	4
2.7	Production of Auxiliary Outputs.....	4
2.8	Production of Run History Information.....	5
2.9	Logic Statement Storage.....	5
2.10	File Update Methods.....	5
2.11	Modes of Operations.....	6
2.12	Transaction Sorting.....	7
2.13	Ordinary Maintenance.....	7
2.14	Checkpoint/Restart.....	8
2.15	Segmented Files.....	9
2.16	Secondary Indexing.....	9
2.17	Auxiliary File Reference.....	10
2.18	Logic Statement Size.....	10
3	FM DESCRIPTION.....	11
# 3.1	Control Elements.....	11
3.2	FM Functioning.....	15
4	INPUTS.....	20
4.1	Card Input.....	20
4.1.1	FMS Control Card.....	20
# 4.1.1.1	LIMIT Control Card.....	20
4.1.2	Segment Control Cards.....	20
4.1.3	Logic Statement Library Update Deck.....	20
4.1.3.1	Library Action Cards.....	21
4.1.3.2	Logic Statement Source Decks.....	22
4.1.3.3	Logic Statement Library Update	

Section		Page
	Terminator Card.....	23
4.2	Transactions.....	23
4.3	Subroutine Library.....	25
4.4	Data File.....	25
5	OUTPUTS.....	26
5.1	Output Data File.....	26
5.2	Auxiliary Output.....	26
5.2.1	Tape and Disk Auxiliary Output.....	26
5.2.2	Punched Card Auxiliary Output.....	26
5.2.3	Printed Auxiliary Output.....	27
5.3	Run History.....	27
# 5.4	File Analysis and Run Optimization Statistics.....	27
6	CONTROL CARD FORMATS.....	30
6.1	Free-Format Specifications.....	30
6.1.1	FMS Control Card (Free-Format).....	31
# 6.1.1.1	LIMIT Control Card (Free Format).....	34
6.1.2	Segment Control Cards.....	34.1
6.1.3	Library Action Card (Free-Format)...	35
6.1.4	Transaction Descriptor (TD) Cards (Free-Format).....	38
6.1.5	Language Identifier Card.....	40
6.1.6	Logic Statement END Card.....	40
6.1.7	Logic Statement Library Update Terminator Card.....	40
6.1.8	Report Identifier Card.....	40
6.2	Ordinary Maintenance (OM) Transaction Descriptor (TD) Cards...	41
6.2.1	Keyword: FIELD.....	42
# 6.2.2	Keyword: CONTROL.....	43
6.2.3	Keyword: PICTURE.....	44
6.2.4	Keyword: VALUE.....	44
6.2.5	Keyword: RANGE.....	45
6.2.6	Keyword: VERIFY.....	46
6.2.7	Keyword: CONVERT.....	46
6.2.8	Keyword: GENERATE.....	47
6.2.9	Keyword: ERROR.....	48
6.3	Fixed-Format Specifications.....	50
6.3.1	FMS Control Card.....	50
6.3.2	Library Action Cards.....	52
6.3.3	Transaction Descriptor (TD) Cards.....	54
6.3.4	Language Identifier Card.....	56
7	ECOL LANGUAGE.....	58
7.1	Card Format.....	58

Section		Page
7.1.1	Symbols.....	58
7.1.2	Operators.....	58
7.1.3	Operands.....	58
7.1.4	Comments.....	58
7.2	Operand Coding.....	59
7.3	POOL Instructions.....	61
7.3.1	Alphabetical Listing.....	61
7.3.2	Valid Operands Chart.....	68
7.3.3	Instruction Groups.....	73
7.4	POOL Instructions.....	80
7.4.1	Environment Handling Instructions...	80
7.4.2	Data Handling Instructions.....	83
7.4.3	Control Instructions.....	92
7.4.4	Display Instructions.....	100
7.4.5	Ordinary Maintenance Validity Test Instructions.....	102
7.4.6	Transaction Error Log Instruction (SCDA and OM).....	103
7.5	Logic Statement Examples.....	104
7.5.1	FMS Control Card.....	104
* 7.5.1.1	LIMIT Control Card.....	104
7.5.2	Library Action Card to Add a Report..	105
7.5.3	Logic Statement Setup.....	105
7.5.4	Use of Data Conversion Subroutines...	109
7.5.5	Periodic Set Processing.....	111
7.5.6	Test for Numeric Data.....	115
7.5.7	Production of Summary Information....	118
7.5.8	Variable Field and Set Processing....	121
7.6	Summary of POOL Instructions.....	125
8	ORDINARY MAINTENANCE (OM) EXAMILES...	128
8.1	Use of Ordinary Maintenance TD Cards.	128
8.2	Use of Ordinary Maintenance TD Cards and POOL Instructions.....	128
9	NEW FILE MAINTENANCE LANGUAGE (NFL).	130
9.1	NFL Statement Composition.....	130
9.1.1	Statement Identifiers.....	131
9.1.2	Keywords.....	132
9.1.3	Noise Words.....	134
9.1.4	Statement Labels.....	134
9.1.5	Operands.....	134.1
9.1.5.1	Control Location Operands.....	135
9.1.5.2	Subroutine/Table Name Operands.....	135
9.1.5.3	Literal Value Operands.....	135
9.1.5.4	Data Location Operands.....	136
9.1.5.4.1	File Data Operands.....	137
9.1.5.4.2	Transaction Data Operands.....	137



Section	Page
9.1.5.4.3 Indirect Data Operands.....	137
9.1.5.4.4 Defined Constant and Area Operands..	138
9.2 Special Requirements and Considerations.....	138
9.2.1 Data Mode Compatibility.....	138
9.2.2 Data Length Compatibility.....	139
9.2.3 Special Statement Sequence Requirements.....	140
9.2.3.1 Condition/Action Statement Sequence..	140
9.2.3.2 Procedure Definitions.....	141
9.2.3.3 Define Sequence Requirements.....	142
9.2.4 Subset Positioning.....	143
9.3 NFL Statement Description.....	143
9.3.1 Conditional Statements.....	143
9.3.1.1 Relational Condition.....	144
9.3.1.2 Table Validation.....	146
9.3.1.3 Picture Mask.....	146
9.3.1.4 Switch Test.....	147
9.3.1.5 Bit Mask Test.....	148
9.3.1.6 New Record Test.....	149
9.3.1.7 Job Complete Test.....	149
9.3.1.8 Overflow Test.....	150
9.3.2 Action Statements.....	150
9.3.2.1 Data Movement.....	150
9.3.2.1.1 The MOVE Statement.....	150
9.3.2.1.2 The ATTACH Statement.....	152
9.3.2.2 The COMPUTE Statement.....	152
9.3.2.3 Subset Positioning Statements.....	154
9.3.2.3.1 The LOCATE Statement.....	154
9.3.2.3.2 The STEP Statement.....	154
9.3.2.3.3 The POSITION Statement.....	155
9.3.2.4 Auxiliary Output Statements.....	157
9.3.2.4.1 The PRINT Statement.....	157
9.3.2.4.2 The FUNCH Statement.....	158
9.3.2.4.3 The WRITE Statement.....	158
9.3.2.4.4 The DISPLAY Statement.....	159
9.3.2.5 The BUILD Statement.....	159
9.3.2.6 The DELETE Statement.....	159
9.3.2.7 The DEFINE Statement.....	161
9.3.2.7.1 Defining a Constant.....	161
9.3.2.7.2 Defining an Interlogic Statement Work Area.....	162
9.3.2.7.3 Defining an Intralogic Statement Work Area.....	163
9.3.2.7.4 Defining and Initializing an Area...	163
9.3.2.8 The TURN Statement.....	164
9.3.2.9 Execution Sequence Changing Statements.....	165

Section	Page
9.3.2.9.1 The GO Statement.....	165
9.3.2.9.2 The RETURN Statement.....	166
9.3.3 Control Point Identifiers.....	166
9.3.3.1 The NOTE Statement.....	166
9.3.3.2 The PROCEDURE Statement.....	167
9.3.3.3 The END Statement.....	167
9.3.3.4 The ELSE Statement.....	167
9.3.3.5 The CONTINUE Statement.....	168
9.3.3.6 The Language Identifier Statement...	168
9.4 NFL Logic Statement Examples.....	169
9.4.1 FMS Control Card.....	169
9.4.2 Library Action Card to Add a Report.	170
9.4.3 Logic Statement Setup.....	170
9.4.4 Use of Data Conversion.....	175
9.4.5 Periodic Set Processing.....	177
9.4.6 Test for Numeric Data.....	180
9.4.7 Production of Summary Information...	182
9.4.8 Variable Field and Variable Set Processing.....	184
9.5 Summary of NFL Condition and Action Statement Syntax.....	186
APPENDIX Utilizing a NIPS File as FM Transaction Input.....	193
DISTRIBUTION.....	195



## FILE MAINTENANCE (FM)

the input file. For details of file block size specification, see Volume VIII, Job Preparation Manual.

\* The FM component features are discussed in the following paragraphs. Another option available to the user is to limit the records read from the data file to be processed. This can be done through the use of a LIMIT statement.

### 2.1 Transaction Sources

The FM component will accept fixed or variable length, blocked or unblocked, or undefined transaction records from tape, disk, or card files. The transaction files may be organized sequentially, or in indexed sequential form on disk. The latter capability permits the FFS data file to be used as transaction input to update another FFS data file.

### 2.2 Transaction Formats

On any given FM run, file updating can be performed with a variety of transaction record formats, from any number of different reporting sources. This capability allows new information management systems to be developed without seriously impacting existing information management systems and existing reporting systems.

### 2.3 Logical Updating and Transaction Editing

The FM component automatically performs all of the I/O functions, record positioning, and new record generation required for a file update run; however, the user supplies the actual record update logic to be used by writing file maintenance logic statements. The user writes one logic statement for each different transaction record format that is used to update his file. The logic statements are written in the File Maintenance languages (POOL or NFL). These languages provide a full complement of comparative and arithmetic functions, so that the user can perform all of his transaction data editing validation in conjunction with his file update. In most applications, no preprocessing or preediting of transaction data should be required.

## FILE MAINTENANCE (FM)

### 2.4 Data Conversion and Validation - User Subroutines

In addition to the comparative and arithmetic commands, the FM languages also provide the user with commands that allow him to perform on-line validation and conversion of transaction data with subroutines that may be written in any of the S/360 languages (CCBOL, FORTRAN, PL1) or in S/360 Assembler language. These subroutines can be prestored on a load library, using the NIPS 360 FFS utility, UTSUBLDR, and can be accessed by the FM component as they are required. A NIPS 360 FFS utility, TABGEN, has been provided to generate tables that perform data conversion and validation using the table search technique (see section 2, Volume VII, Utility Support.)

### 2.5 Processing of Periodic Sets

The FM languages contain a series of commands that allow the user to scan the periodic sets, so that he may selectively update subsets based on their existing data contents. These commands also allow the user to control the physical sequence of subsets in a set when new subsets are being added.

### 2.6 Variable Field and Variable Set Maintenance

The FM languages provide a set of commands for adding, deleting, or replacing information in the variable fields and the variable sets.

### 2.7 Production of Auxiliary Outputs

The FM languages provide a set of commands for producing auxiliary outputs on printer, punch, direct access device or tape in conjunction with a file update. These outputs can be used to provide an audit trail of the updates or to produce transaction files that might be used to update other files in a file system. Two separate printed reports, two separate punched outputs, and up to five auxiliary files may be produced. The user has access to a 999-byte work area for formatting auxiliary output records. Each record is limited to a maximum of 994 bytes because NIPS adds six overhead bytes to the data and the DCB LRECL is 1000. Violation of this limit will cause a System 001 ABEND (I/O ERROR, RECORD TOO LONG). Information can be selected for

## FILE MAINTENANCE (FM)

output from either the transaction records or the data records. The FM languages also provide the capability for maintaining summary counts and totals which can be output to tape, print, or punch. The user has access to 20 full-word binary work areas for developing summary information in conjunction with a file update run.

### 2.8 Production of Run History Information

The FM component automatically produces a printed run history to indicate error conditions that were encountered during file updating. The run history is printed separately from the printed auxiliary output.

### 2.9 Logic Statement Storage

The FM component maintains a library of the user-written logic statements (in compiled form) to update a file. The Logic Statement Library is maintained with the data file and consists of a series of records with special keys. This capability allows the user to avoid recompilation of the statements each time his file is to be updated, thereby reducing execution time. The component automatically retrieves the logic statements that are required for the transaction record formats encountered during an update cycle. The user can add/delete statements on the Logic Statement library by using a set of control cards, called library action cards (see section 4.1.2).

### 2.17 File Update Methods

The FM user can cause the FM component to employ any combination of the following file update methods in a given file maintenance run.

- a. Range Updating - Using this method, every record of the data file is made available for logical updating. This method may be used to make corrections to some data field in the file or to produce summary information. If at all possible, this method should not be used as a part of normal production runs since it greatly reduces the efficiency of the component. However, when it is deemed necessary, the RANGE statement should be compiled and stored on the library in accordance



## FILE MAINTENANCE (FM)

with the paragraph "Logic Statements" under section 3.1, Control Elements.

- b. Exception Updating - Using this method, each transaction record is matched against a particular data record, which is made available for logical updating. When no match is found, a new data record is generated, and a 'new record' indicator is set which the user can test by using one of the FM language commands in his logic statement. Exception updating is the normal update method.
- c. Exception-Range Updating - Using this method, each record that has been subjected to exception updating is made available for further processing by one or more exception-range logic statements. This method can be used for collecting summary information or for producing audit information on all records affected by an update cycle.
- d. Direct Subset Updating - Using this method, a transaction record is matched against a particular data record subset and the record's fixed set, and the particular subset is made available for processing. When no subset match is found, a new subset is generated, unless the record does not exist. In the latter case, the transaction is logged as an error on the run history.

### 2.11 Modes of Operations

Four FM component modes of operation may be specified by the user. (Any given FM run may be executed in only one mode of operation.)

Compile Logic Statement Mode - This mode allows the user to get compilations of his logic statement for the purpose of debugging. The component produces a symbolic listing with all errors flagged, along with a summary of the errors and their causes. Processing is completed when the last statement is compiled. No logic statements can be added to the File Library portion of the data file in this mode, but any other actions can be performed on the Logic Statement Library. If the data file is sequentially organized, no



## FILE MAINTENANCE (FM)

library actions will be performed on the Logic Statement Library.

**Logic Statement Library Update Mode** - When this mode of operation is specified, FM updates the Logic Statement Library as specified by the library action cards. Logic statements may also be compiled in this mode and placed on the Logic Statement Library. If the data file is sequentially organized, logic statements cannot be added to the logic statement portion of the data file during this mode.

**Data File Generation Mode** - The user specifies this mode of operation when he wishes to generate a new file. The component will also compile logic statements and perform Logic Statement Library maintenance if specified. At this time, the user may specify the file block size using the BSZNEWF symbolic parameter of the XFM procedure. If a block size is not specified, it will be set at the size of the input FFT.

**Data File Update Mode** - This mode of operation is specified when the user wishes to update a data file. Logic statement compilation and library maintenance may also be performed by the component when executed in this mode of operation. If the file is sequentially organized, the user may specify a new file block size using the BSZNEWF symbolic parameter in the XFM procedure. If a new block size is not specified, it will be set at the size of the input file.

### 2.12 Transaction Sorting

The FM component automatically sorts all transaction records into sequence on the transaction control fields prior to updating a file. The sort will be accomplished on tape or disk. When the transaction volume is such that the disk sort capacity is exceeded, a tape sort will be executed. If the transactions are already in order, the sort is automatically bypassed.

### 2.13 Ordinary Maintenance

Through the use of the Ordinary Maintenance (OM) Transaction Descriptor (TD) cards (see section 6.2), the

## FILE MAINTENANCE (FM)

user may specify automatic validation of transaction data, and automatic (unconditional) updating of file fields. The types of data validation that may be performed are value, range, picture, and verification against a table or subroutine. Data conversion through use of table or subroutine is also permitted. Error options are provided to permit or suppress automatic logging of erroneous data, to automatically delete records of subsets generated by the erroneous transaction data, or to automatically clear the data field that corresponds to the erroneous transaction field.

The Ordinary Maintenance capability permits the user to write logic statements that contain only Ordinary Maintenance Transaction Descriptor cards, or to write statements that contain both Ordinary Maintenance cards and FM statements. Instructions are provided in the FM language to interrogate the results of the Ordinary Maintenance validation (see section 7.4.5). This allows the user to perform part of the validation function using the Ordinary Maintenance language, and to perform the more complex edits and data manipulation in the FM language in one logic statement. In a mixed logic statement, the Ordinary Maintenance functions are executed prior to the FM language functions.

### 2.14 Checkpoint/Restart

During the generate/update phases of SAM processing, the user may invoke the OS/360 checkpoint/restart capability to record timed or end-of-volume checkpoints. End-of-volume checkpoints will be taken on the SAM data file input for an FM SAM update and on the user's SAM transaction input, if there is one, for a generate or an update. The checkpoint/restart capability should be used only during long-running jobs using the execute only procedures. (Note that OS/360 step restart is program-independent and is not the topic of this discussion. A detailed description of the OS/360 checkpoint/restart capability, which is utilized in NIPS, is available to the interested user in IBM Systems Reference Library, Number C28-6708.) A detailed description of how to use checkpoint/restart in NIPS is included in the Job Preparation Manual, Volume VIII.

## FILE MAINTENANCE (FM)

### Section 3

#### FM DESCRIPTION

In addition to the FM logic statements, the FM component functions under the control of information that exists in the user-supplied control cards and in the transaction and data file records. The function and interaction of these elements is discussed in the following paragraphs.

#### 3.1 Control Elements

The FM component operates under the control of the following elements of information: FMS control card, LIMIT control card transactions, and logic statements.

- a. FMS Control Card - The user must provide an FMS control card (see subsection 6.1.1 and 6.3.1) for each FM run. It specifies the functions that the FM is to perform.

- #
  - o LIMIT Control Card - The user may optionally provide a LIMIT card (see subsection 4.1.1.1 and 6.1.1.1). It specifies the range of records to be processed.

- b. Transactions - Transaction records are source-data records containing information used to update the data records. The transaction records may also contain two elements of control information. They are the transaction ID fields and the transaction control fields.

- o Transaction ID Fields - Since more than one type of transaction record can be used to update a given data file, each transaction record must contain information that uniquely identifies its format. The fields that contain this identifying information are the



## FILE MAINTENANCE (FM)

transaction ID fields. Each transaction may contain up to three ID fields. The aggregate length of the transaction ID fields may not exceed six characters.

- o Transaction Control Fields - If the data in a transaction record is to be applied to a particular data file record or a record subset, that transaction must contain control fields. The most significant information contained in these fields is identical to the information contained in the control fields of the data record to which the transaction applies. The least significant information in these fields may be used to control the sequence in which different transactions are processed against a given data record and must not exceed 10 characters. The user specifies the location of transaction control fields through the use of transaction description cards, which are a part of the logic statements. A transaction record may contain up to 255 characters of control information which may include up to 10 characters of user control information in as many as 60 noncontiguous fields.

- o Transaction Report Identification - Each transaction that is used to update a file contains a transaction ID. The location of the transaction ID fields for all of the different transactions within a given report must be the same. To identify the location of the transaction ID fields, the user must assign a report name to each set of transactions with ID fields in unique locations. The report name must conform to the system name rules specified in Volume I, Introduction to File Concepts. Before FM will compile the logic statements that are used in processing a given set of transactions, the transaction report identification record must be placed on the Logic Statement Library through the use of a library action card.



## FILE MAINTENANCE (FM)

Transactions that are being processed during an FM run must be identified by a report name before they can be processed through the use of the FMS control card and report identification card. Transactions from several different reports may be used to update a file on a given run. Report identification cards/records must be inserted in the input stream between groups of transactions from different reports.

- c. Logic Statements - In addition to specifying the processing logic, the analyst must also include certain control information in his logic statements. This information is used to specify the type of logic statement he is constructing and to identify the logic statement so that it may be retrieved from the library.

There are six types of logic statements:

- o RANGE Statements without Transaction Data - RANGE statements without transaction data are used to perform logical processing on every record in the data file. The analyst specifies this type of statement by omitting transaction information when he makes the library action cards for his logic statement. This type of statement cannot be stored and must be recompiled for each FM execution. It should be noted that this type of statement can easily be converted to a range with transaction data in accordance with the paragraph below to permit storing on the library. A single transaction description card containing a noncontrol field (i.e., \$ISNAM,1,1) with the appropriate ASP card would fulfill the requirements for storage of a logic statement. A single transaction with the applicable transaction control ID columns filled in would subsequently invoke the logic statement at generate/update time (e.g., an A in the first position of a transaction).

## FILE MAINTENANCE (FM)

- o RANGE Statements with Transaction Data - RANGE statements with transaction data are used to perform logical processing on every record in the data file and to update the record. The analyst specifies this type of statement by not specifying transaction control fields when he makes up transaction description cards for his logic statement. This type of statement may be stored on the Logic Statement Library and need not be recompiled for later use.

RANGE statements apply to the entire data file while EXCEPTION statements apply to one specific record. These types of updating operations may be combined.

- o EXCEPTION Statements - EXCEPTION statements always require transaction data and are used to process a particular record of a data file. The transaction control fields indicate which record is to be processed. The analyst specifies this type of statement by specifying the location of the transaction control fields in the transaction description cards. EXCEPTION statements may be stored on the Logic Statement Library and need not be recompiled for later use.
- o SUBSET EXCEPTION Statements - SUBSET EXCEPTION statements process a particular subset record of a data file. The analyst specifies this type of statement by specifying a subset field as a minor control field in his transaction description cards. The subset ID must be unique within its set. The transaction control fields, major and minor, indicate the subset to be processed. This type of statement may be stored and need not be recompiled for later use.
- o EXCEPTION RANGE Statements - EXCEPTION RANGE statements may be with or without transaction data. The statements are used on a per-record basis to process all data records in the file that have been updated via EXCEPTION

## FILE MAINTENANCE (FM)

statements during a given run. The analyst specifies this type of statement by using the POOL XNP instruction written in the FOOL language logic statement, or by punching XNP as the sixth parameter in a free format Add Statement card (subsection 6.1.2) when using Ordinary Maintenance or the NFI. These statements can be stored on the Logic Statement Library when they are associated with transaction data.

- o Logic Statement Identification - All logic statements except RANGE and EXCEPTCN RANGE statements without transaction data, must be uniquely identified within a data file. Statement identification is accomplished through the use of the library action card that must precede each logic statement compiled. The report name and the statement must be specified.

The report name identifies the report type and provides the location of the transaction ID field for a set of transaction records; this is the higher level of identification.

The statement name must be identical to the transaction ID on the transactions used with the logic statement. This is the lower level of identification and must be unique within a given report.

### 3.2 FM Functioning

The FM component is divided into the following functional sections:

- a. Initialization - This section processes the user's punched card input. It uses the FMS control card to determine which FM functions are to be performed and sets up a run communication record to control processing for the run. It will also process any segment control cards and update the segment records on the data file. It constructs a disk work file from the user's Logic Statement Library



## FILE MAINTENANCE (FM)

update cards; if there are any card transactions, these are output to a second work file.

- b. Logic Statement Compilation and Library Maintenance - The functions of this section consist of deleting specified reports and statements from the Logic Statement Library, adding new reports to the library, and compiling and adding new statements to the library for a given file. The report information records and the logic statements are maintained on the library in sequence by report and statement name. Each logic statement on the library contains two parts. The first part is a logic statement control record that is used during transaction processing. This record indicates the statement type. If it is an EXCEPTION or SUBSET EXCEPTION statement, it indicates the location of the transaction control fields in the transaction record it processes. The second part consists of the Executable Load module that is produced when the POOL language or NFL statements are compiled.

In addition to updating the Logic Statement Library, this section also produces a listing of the user's Logic Statement Library update deck, with any errors flagged. When the compile-only mode of FM is specified, the Logic Statement Library is only updated by the addition or the deletion of reports/statements. For compiled statements, only the error listing is produced.

- c. Transaction Processing - The Transaction Processor matches each transaction record with its appropriate logic statement and creates an update record for each transaction that is used by the File Processing section to perform the actual record update.

After the Transaction Processor reads a transaction record, it extracts the transaction ID fields from the record. The locations of the transaction ID fields are associated with the transaction report name on the Logic Statement Library. It then uses the transaction report name and the transaction ID



## FILE MAINTENANCE (FM)

to retrieve the control record of the logic statement that will be used to process the transaction.

An update record, containing the name of the logic statement that processed the transaction, a sort key, and the transaction record, are then produced. The high-order byte of the sort key contains the update record type indicator; 'H' for Range updates, 'E' for Exception Range updates, and 'P' for Exception and Direct Subset updates. The collating values of the record-type indicators establish the sequence in which the update records will be passed to the File Processing section, but do not establish the sequence in which the update records will actually be processed. For Exception update records, the sort key will contain only the major ID, one byte of binary zeros for the set number and user control information. For direct subset updates, the sort key will contain the major ID, the set number and the subset control field. The remainder of the sort field will be padded with binary zeros.

At the completion of transaction processing, the update records are ordered on the sort key, if they are not already in sequence.

If the data file is a segmented file and segment processing is to be performed, the update record sort keys are checked to determine if the record key is within the boundaries of the segment being processed. If it is not, an error message will be printed and the update record will be bypassed.

d. File Processing - The Range and the Exception Range update records are read and saved in core. Then processing of the Exception and Direct Subset update records begins. Direct Subset updating for a given record will be performed following the exception updates on the record. If only Direct Subset updating is to be performed on the record, the fixed set and the subset are retrieved, and the updating is performed. If no matching fixed set is found, an error message is logged. When a matching

## FILE MAINTENANCE (FM)

subset is not found, a new subset is generated, and a 'NEW RECORD' switch is set that can be tested by the user with the POOL 'BNR' instruction or the NFL new record test.

If Range updates are to be made, the entire data file is passed sequentially. As each data file record is read, its record key is matched against the current update record's sort key. If a match is found, the appropriate logic statement is executed, and the current data record is updated with the information in the current update record. A new update record is then read and processing continues on the same data record, until an update record is read with a different sort key. At this point, Range processing is executed against the current record.

If the current update record's sort key has a lower value than the current data record's key, a new record is generated, and the new record switch is set.

If the current data record's key has a lower value than the current update record's sort key, this indicates that no exception processing is to be performed against the data record, and only the range processing is performed.

When Range processing is performed against the records that were not updated by exception processing, the Exception Range logic statements are not executed.

When all of the data records and update records have been processed, the RANGE statements and EXCEPTION RANGE statements that have been collecting summary information are executed once more, to allow them to output the information.

When no Range processing is required, the File Processing Section retrieves and processes only those records with keys that match the Exception update record sort keys. When no match is found, a new record is generated.

## FILE MAINTENANCE (FM)

File generation is accomplished by generating a new record for each Exception update record with a unique sort key.

Subset Exception updates can be processed during file generation for those records generated by Exception update records.

For sequential processing, the file is passed sequentially as for Range updating. The FFT and logic statement records are copied onto a new sequential output data file. The records are read and updated, and the records whose major IDs have been changed are written on a temporary hold file. This hold file is sorted and merged with the output data file to produce the new data file. If the sequential file is a segmented file, a listing of the segment control records on the data file will be printed indicating segment boundaries and the volume serial number of each segment. During logic statement execution, if a major control field change occurs, the new Record ID is checked to insure that the new ID is within the segment boundary. If the new Record ID is not within the segment boundary, an error message will be printed and the change will not be executed.

A secondary function of the File Processing section is the production of a consolidated auxiliary output file. The records for this file are produced when any of the POOL or NFL instructions that produce printer, punched card, or tape auxiliary output are executed. Control is then passed to the next section.

- e. Auxiliary Output Processing - This phase reads the consolidated auxiliary output file, and outputs the individual records to their proper printer, punched card, disk or tape files. When two printer files are requested, they are produced sequentially on the same printer. The two punched card files are punched into pockets 1 and 2 of the card punch.



## FILE MAINTENANCE (FM)

### Section 4

#### INPUTS

##### 4.1 Card Input

Card input to the FM component consists of the FMS control card, the Logic Statement Library update deck, and the transaction deck.

##### 4.1.1 FMS Control Card

The format of the FMS control card is described in sections 6.1.1 (Free-Format) and 6.3.1 (Fixed-Format).

An FMS control card must be the first card in the FM run deck. It specifies the functions to be performed during the FM run.

##### #4.1.1.1 LIMIT Control Card

The format of the LIMIT control card is described in section 6.1.1.1 (Free Format). The LIMIT control card is optional. When user, it must follow the FMS control card. It specifies the range of records that File Maintenance will process.

##### 4.1.2 Segment Control Cards

The segment control cards will direct the component to perform processing of the segment records on the data file. The segment control cards must appear immediately after the FMS control card. If segmented processing is not desired, these cards must be omitted. The format of these cards is described in section 6.1.2.

##### 4.1.3 Logic Statement Library Update Deck

This deck specifies the maintenance actions to be performed on the Logic Statement Library and consists of the



FILE MAINTENANCE (FM)

cards discussed. If no Logic Statement Library update is desired on a given run, there is no requirement for this deck.

## FILE MAINTENANCE (FM)

### 5.2.3 Printed Auxiliary Output

The FM component provides the user with two printed outputs. The user is responsible for formatting the print lines. The FM component will handle the printing of 132 characters or less. If the user requests that more than 132 characters of data be printed, the data in excess of 132 characters will be printed on subsequent lines.

### 5.3 Run History

As part of its functioning, the FM component automatically generates a run history on the printer. This includes listings of logic statements that are compiled, and messages indicating that errors, or unusual conditions that might be interpreted as errors, have been encountered during processing.

If segmented file processing is being performed, the segment control records on the current segment will be printed showing the segment boundary and the volume serial number of each segment.

### 5.4 File Analysis and Run Optimization Statistics

\* The File Analysis Statistics capability in the FM component provides transactions showing the number of times each logic statement is executed during an FM execution. The data set (DSNAME) of this data set must be the data file name suffixed by a T. The T is added to ISAM 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 track 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.

## FILE MAINTENANCE (FM)

\* If the user specifies a DSNNAME (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 Run Optimization Statistics capability provides the user with statistical data reflecting the core allocation during FM execution. The breakdown of the statistics detail the amount of core used for user subroutines and tables, logic statements, process blocks, I/O buffers, and access methods. It also includes the number of BLDL entries allocated and used and the number of entries required for each subroutine, table, and logic statement to reside in core. The amount of core required for each subroutine, table, and logic statement to reside in core is also output. If subroutines, tables, and logic statements are rolled, this information will be output with the causes for the rolling and the number of times it occurred.

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

The statistics gathering is initiated through parameters entered in the PARM field of the EXEC card. The parameters and their functions are as follows:

- ROS - Indicates that run optimization information is to be gathered and output.
- NOROS - Indicates that optimization processing is to be omitted. If no other parameters are coded, this parameter should be omitted as it is the default.

The parameters the user may supply to tailor his core allocation are listed below. Using these parameters, the Run Optimization Statistics are gathered and output unless the NOROS parameter is used.

- TCP=NK - The N indicates the number of bytes requested for the process block in 1000 (K) bytes.



## FILE MAINTENANCE (FM)

- TCB=n        -     The n indicates the number of entries to be used in the BLDL list for SUBSUF, the subroutine supervisor.
- TCS        -     This parameter indicates that the statistics record on the ISAM data file is to be used to compute the process block size. This parameter must not be used with TCP and vice versa.

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

## FILE MAINTENANCE (FM)

### Section 6

#### CONTROL CARD FORMATS

##### 6.1 Free-Format Specifications

This section specifies the preparation requirements for all FM control cards. These cards may be punched in free format or fixed format (see sections 6.1 and 6.3 respectively).

The general rules that apply to free-format control cards are as follows:

- a. The control card data must always be punched starting in column 1. The first character of a control card must always be a dollar sign (\$).
- b. The information in the cards must be punched in a specified parameter sequence.
- c. The control card fields must be separated by commas, with no intervening blanks.
- d. If the analyst has no requirement for a certain parameter, he must so indicate by punching a comma for that field, except when he has no more fields to punch.

# The four control cards that may be formatted in this manner are as follows:

- a. The FMS control card
- # b. The LIMIT control card
- c. The Library Action card
- d. The Transaction Descriptor card.

## FILE MAINTENANCE (FM)

The data file type parameters are:

TAPE - For sequential processing (SAM)

DISK - For indexed sequential processing (ISAM)

The default option is to process, according to the organization of the input data file which contains the FFT, logic statements and, for UPD runs, data records.

Field 6 - Transaction Source (Required for UPD or GEN Mode)

TAPE - Sequential transactions; file on either tape or direct access storage.

DISK - Transaction source is in indexed sequential organization

SAM - Transaction source is a NIPS 360 FFS data file in sequential organization

ISAM - Transaction source is a NIPS 360 FFS data file in index sequential organization

CARD - CARD must be specified or this parameter omitted entirely when utilizing multiple transaction sources since the NEW FREPORT card describing the source must be included in the run deck.

NONE - No transactions

If this parameter is omitted, CARD is assumed.

Field 7 - Segmented File Processing Indicator

SEG - Segmented processing to be performed in this run

NOSEG - Bypass segmented file processing in this run



## FILE MAINTENANCE (FM)

If this parameter is omitted, the default option is to perform segmented file processing if the data file is a segment and to ignore segmented file processing if the file is not a segment.

### #6.1.1.1 LIMIT Control Card

#### Description:

Field 1 - \$LIMIT - card identifier

Field 2 - Field name or Group name [m/n] [#SUBTAB]

The field or group name specified must include the highorder character(s) of the major control field. The user has the option to specify partial field notation for the field or group by indicating m/n. This specifies which portions of the record key will be used for comparison. This partial field must start at the first character; i.e. 1/n. In addition, the field or group may be modified by a subroutine expression. Double pound signs (##) suppress automatic table conversion, and the name of the subroutine enclosed in pound signs forces table conversion.

Field 3 - Relational Operator

The relational operators shown below are allowed in the statement formed by the LIMIT operator and condition the selection of records as follows:

EQ - process when equal to  
LT - process when less than  
LE - process when less than or equal to  
GT - process when greater than  
GE - process when equal to greater than  
BT - process when equal to or between

The logical connector NOT may precede all relational operators.

## FILE MAINTENANCE (FM)

Field 4 - Literal(s)

If the BT relational operator is specified, then two literals are required and must be separated by a slash; i.e., a/b.

### 6.1.2 Segment Control Cards

The segment control cards are used to update the segment records on a segmented data file. The options allowed are as follows:

- SEG - This option indicates to the component that the output of a GEN run is to be a segmented data file. This option must be used only when the mode of the run is GEN.
- ADD - This option indicates to the component that a new segment record is to be added to the segmented data file. This option may be used in either GEN or UPD mode.
- REP - This option indicates to the component that the volume serial number of a specified segment is to be replaced by a new volume serial number.
- DEL - This option indicates to the component that the segment record specified by the lcw key value is to be deleted from the segmented data file.

Note: The actions specified by the segment control cards will be performed at FM initialization time. Therefore, if no logic statements are to be compiled, omit the logic statement parameter on the FMS control card.

The control cards are free format and possible operands are as follows:

<u>\$SEG</u>	LCKEY HIKEY	
<u>\$ADD</u>	LCKEY HIKEY	VOLID
<u>\$REP</u>	LCKEY CVOID	NVOLID

## FILE MAINTENANCE (FM)

- d. Field notation indicator (optional). This field consists of a one-byte code to indicate the type of data contained in the transaction field. The legal codes and their meanings are as follows:

- 'A' - The field contains alphanumeric data, including blanks and special characters.
- 'B' - The field contains numeric data in binary format.
- 'C' - The field contains coordinate data in internal form.
- 'D' - The field contains numeric data in decimal format.

The default option for this code is 'D'. Decimal ('D') transaction data may be processed in the FOOL language by either the arithmetic (MNU, MNC, CCN) or logical (MAL, MAC, COA) instructions.

6.2.2 Keyword: CONTROL

Abbreviation: CTL

Example: CCNTRCL PSCTL

Function:

# This keyword is used to specify that a transaction field is a control field. The operand may be a one- to two-digit number, to indicate that the transaction field is a major or record control field; or it may be a subset control field mnemonic, which indicates that the transaction field is a subset control field for a direct subset update statement.

For main or record control fields, the number indicates the sequence in which the transaction fields are to be arranged in order to compare them to the data record ID. The transaction fields assigned the lower sequence numbers that have an aggregate length equal to the length of the major data record control group constitute the major transaction control fields; the remaining transaction



## FILE MAINTENANCE (FM) •

control fields are considered user control fields, and are used only to control the update sequence on a given record.

When the control parameter is specified, it must immediately follow the field parameter list.

### 6.2.3 Keyword: PICTURE

Abbreviation: PIC

Example: PIC AABBB\*S (A)NN A(AB)N

Function:

This keyword is used to indicate that character or profile checks are to be performed on a transaction field. The keyword should be followed by masks depicting the types of EBCDIC characters permitted in the defined transaction field. The PICTURE parameter may only be specified for alphanumeric or decimal transaction fields. Character checks that may be specified by the masks are Alphabetic (A), Numeric (N), Special (S), Blank (B), Nonblank (X), Non-special (Y), and no check or universal match (\*).

A direct test for specific characters, as opposed to types of characters, may be specified with a VALUE check, or by enclosing the specific characters in parentheses. The picture masks, not counting parentheses, must be either shorter than or equal to the length of the transaction field defined. If a mask is shorter, only the leftmost characters of the transaction field will be checked, up to the length of the mask. The picture mask is terminated by the occurrence of a blank. Up to 10 masks may be specified in a PICTURE parameter list.

### 6.2.4 Keyword: VALUE

Abbreviation: VAL

Example: VAL 32768 8192 \*\*\*58 'AABX3'

## FILE MAINTENANCE (FM)

Maintenance error tests. The transaction field must be designated by its assigned TD mnemonic.

### Branch on Transaction Valid (Group 3)

BTV        A

This instruction branches to location 'A' if all of the transaction data passed on specified Ordinary Maintenance validity tests.

### Branch on Transaction Not Valid (Group 3)

BTN        A

This instruction branches to location 'A' if any transaction fields failed the specified validity tests.

## 7.4.6 Transaction Error Log Instruction (SODA and CM)

### Log Erroneous Transaction Data (Group 23)

ERR        A,E

Example -        ERR    \$TRANS, 'THIS IS BAD DATA'

This instruction is provided to assist the terminal operator in correcting erroneous transaction data when using the on-line update capability, Source Data Automation (SODA).

When this instruction is executed during a SODA run, it causes the displayed transaction field 'A' to be underscored with a key to the message provided as the literal in field 'B'. (See the Terminal Processing (TP) component volume of the NIPS 360 PFS Users Manual.)

When this instruction is executed during a batch FM run, it causes the message to be printed on the Ordinary Maintenance error log (see section 6.2). In a combined Ordinary Maintenance/ECCL logic statement, this instruction may be used to replace or supplement the standard OM error messages.

## FILE MAINTENANCE (FM)

### 7.5 Logic Statement Examples

The following examples illustrate the setup of the FM run deck for updating the Logic Statement Library, and the use of some of the POOL language instructions. Sections 9.5.3 through 9.5.8 provide equivalent NFL logic statements for the POOL language statements of sections 7.5.3 through 7.5.8. All of the examples pertain to the TEST360 file. For a description of this file, see the Introduction to File Concepts volume of the NIPS 360 FFS Users Manual.

All of the sample logic statements, with the exception of the Range statement, perform updates with transactions from the report 'RPT360'. The different transaction formats within this report are identified by the letters 'A' through 'G' in column 1 of the transaction.

Sections 7.5.1 through 7.5.8 illustrate the free-format FMS control card, library action cards, and TD cards.

Comments cards are shown in each of the logic statements and explain the functions of the statements.

Section 8.1 provides an equivalent Ordinary Maintenance TD of the POCI statement in section 7.5.6. Section 8.2 shows a combination OM/POOL logic statement.

#### 7.5.1 FMS Control Card

The following FMS control card would be used to execute the 'LIB' mode of FM, to perform updates for the logic statement library for the TEST360 file:

```
$FMS/LIB,TEST360
```

##### #7.5.1.1 LIMIT Control Card

The following card would be used when the user wanted to limit a range to all the records whose control field, LCTRL, was equal to 'AAA'.

```
$LIMIT,LCTRL,EQ,AAA  
$LIMIT,LCTRL,BT,AAA/AAA
```

or



## FILE MAINTENANCE (FM)

If all records not equal to 'AAA' were desired, then the control card would be:

```
$LIMIT,ICTEL,NOT,EQ,AAA
```

### 7.5.2 Library Action Card to Add a Report

The following card would be used to add the report 'RPT360' to the Logic Statement Library. The transaction ID field, for transactions within this report, is located in column 1.

```
$AR,RPT360,1
```

This card could also have been punched as follows:

```
$AR,RPT360,1-1
```

However, since the transaction ID field is only one byte long, the '-1' is not required.

### 7.5.3 Logic Statement Setup

The following example illustrates the organization of the library action card, the ID cards, the language identifier card, and the POOL instruction cards for an Exception logic statement. The sample statement performs updates with the 'A' transaction format of the report 'RPT360'.

The first card for the statement is the library action card. This card specifies that the statement is to be permanently added to the library. It also specifies that the statement will perform updates with the 'A' transaction of report 'RPT360', and that the fixed data in that transaction format is 82 bytes long. The transaction does not contain any variable data.

The ID cards follow the library action card. The first field in each of these cards is used to assign mnemonics to the transaction data fields.

## FILE MAINTENANCE (FM)

The second and third fields specify the high-order position and the low-order position of the transaction fields.

The fourth field is used to specify that a transaction field is a major or user control field. In the example, \$RECID is a major transaction control field, and it corresponds to the data record control group, 'UIC'. \$SORT is a user transaction control field. It is not used in matching a transaction record to a data record, but is associated with the record control field to control the file processing sequence.

The fifth field in the TD card indicates the type of data that the transaction fields will contain. The 'A' transaction contains alphabetic (A) data and zoned decimal (D) data only. Insertion of this field is optional, with the default option being 'D'.

The card following the TD card is the language identifier card, and contains the word 'POCL' in columns 16-19.

Comments cards, describing the logic statement's function, follow the language identifier card. The comments cards are identified by an asterisk (\*) in column 6.

The logic statement first tests the new record switch by using the BNR instruction. A new record will be generated by FM when no data record can be found with a UIC group that matches the contents the \$RECID transaction field, in an 'A' transaction. When this occurs, the instructions at NEWREC will be executed. These instructions format a print line in the EBCDIC work area and print the line. At the completion of this function, the instruction sequence at MOVE is executed. This sequence of instructions moves the data from the transaction fields to the data record fields, using the MAL, MAC, and MNC instructions. The MAL and MAC instructions are used to move the alphabetic data to the file record, and the MNC instruction is used to move the numeric data to the file record.

When the MAC instruction at MOVE is executed, no data transfer will take place if the content of the transaction field \$HOME is blank. When the MNC instruction is executed,

## FILE MAINTENANCE (FM)

no data transfer will take place if the content of the transaction field \$FEES is blank.

The instructions that move the transaction data to the coordinate fields DAPI1-4 will automatically convert the data to internal coordinate format.

At the completion of execution of the data move instructions, an SDI instruction is executed. This instruction stores the date/time of the update in the data field LAUD.

A HLT instruction is executed next. This instruction causes an exit from the logic statement.