HEWLETT hp PACKARD

REAL TIME EXECUTIVE BATCH/SPOOL MONITOR

STUDENT WORKBOOK

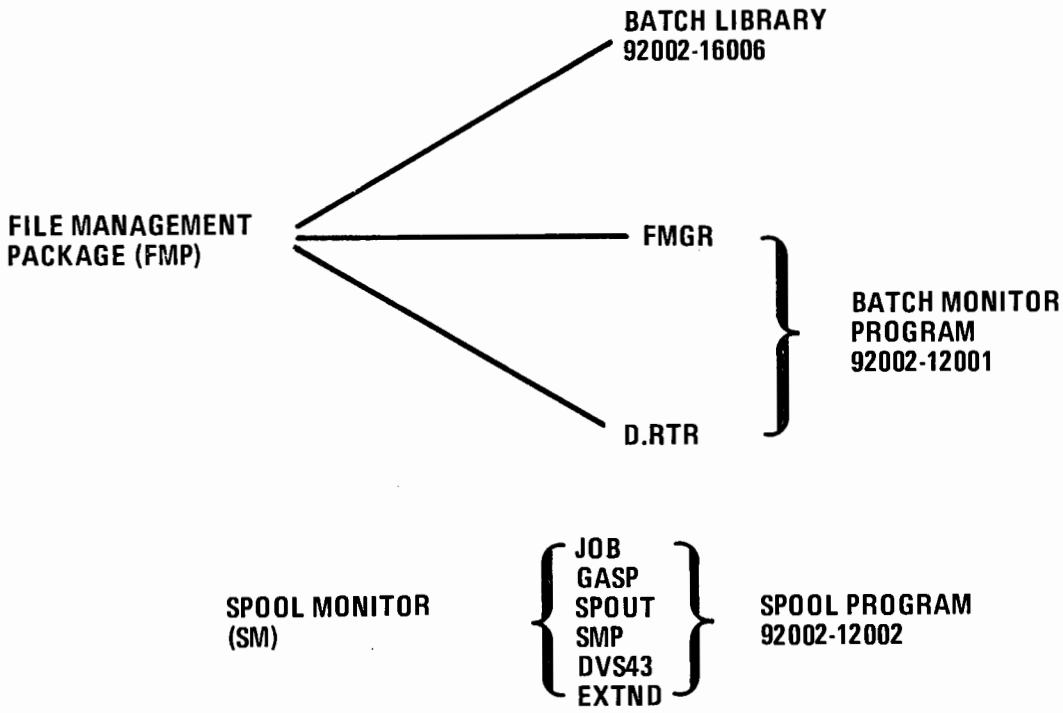
systems engineering

22999-90026

Revised June 1975

INTRODUCTION

"BATCH-SPOOL MONITOR"



USE OF SPOOL MONITOR PROGRAM IS OPTIONAL.

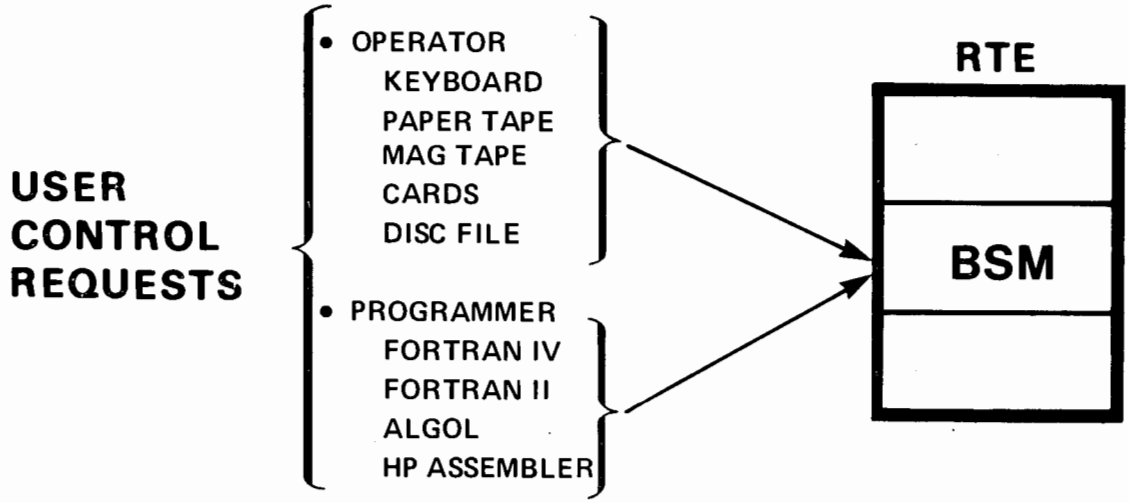
BSM FUNCTIONS

- FILE MANAGEMENT
- SIMPLIFIED PROGRAM DEVELOPMENT FEATURING:

  GENERALIZED PROCEDURES
- BATCHED JOBS WITHOUT SPOOLING
- BATCHED JOBS WITH SPOOLING FEATURING:

  LU TO FILE EQUIVALENCE

BSM ACCESS



THE USER MAY ACCESS THE BSM ON-LINE USING THE OPERATOR COMMANDS OR THROUGH A RTE PROGRAM USING THE PROGRAM COMMANDS

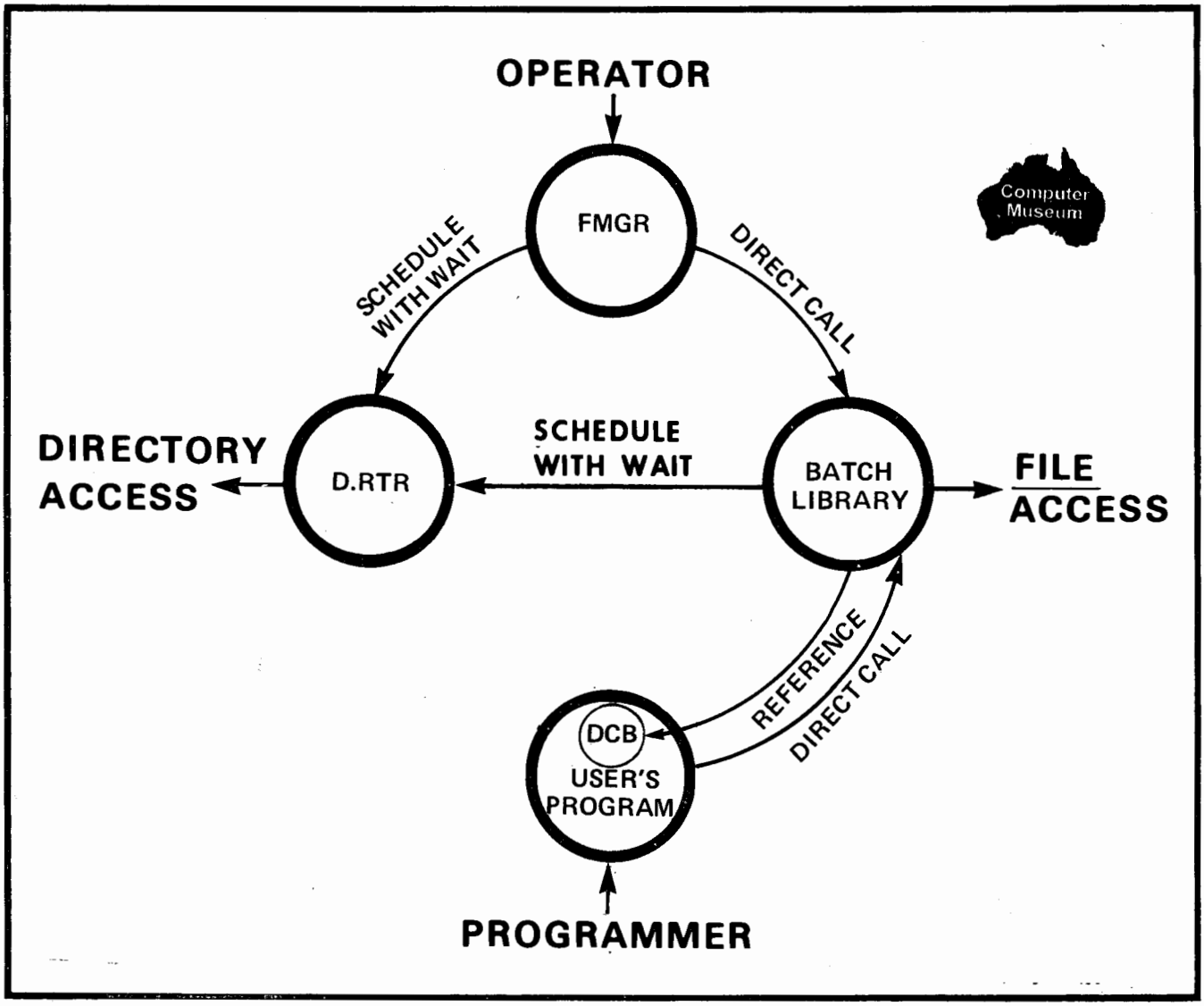
HARDWARE REQUIREMENTS AND PHYSICAL ARRANGEMENT

THE RTE BATCH/SPOOL MONITOR OPERATES WITHIN THE RTE HARDWARE ENVIRONMENT:

RTE-II WITH MOVING HEAD DISC

- FILES MAY EXIST ON CONTIGUOUS TRACKS ON ONE OR MORE DISCS.
- FILES MAY EXIST ON SYSTEM AND AUXILIARY AND ON PERIPHERAL DISCS. UP TO 31 DISCS CAN BE ACTIVE AT A GIVEN TIME.
- A FILE MAY BE DIRECTED TO A PREFERRED DISC BY THE USER, OR
- FMP WILL DIRECT THE FILE TO A CONVENIENT LOCATION.

OPERATOR COMMANDS-
FILE MANAGEMENT



INTER-RELATION BETWEEN FILE MANAGEMENT MODULES OF THE FMP.

FMP OPERATOR INTERFACE

- THE OPERATOR GAINS ATTENTION OF THE FMGR AS FOLLOWS:

  *ON, FMGR (, OPTIONAL PARAMETERS)

  :DL

- THIS SCHEDULES FMGR WHICH RETURNS THE COLON (:)
- THE OPERATOR MAY THEN ENTER ANY OF THE FMP COMMANDS

WHEN FMGR IS SCHEDULED IT ASSUMES THE COMMAND INPUT DEVICE WILL BE THE SYSTEM CONSOLE KEYBOARD. IF THE DEVICE IS NOT A KEYBOARD, COMMANDS READ FROM THE INPUT DEVICE (SUCH AS A TAPE READER) MUST BE PROCEEDED BY A COLON (:).

FMGR MAY ALSO BE SCHEDULED BY A USER'S PROGRAM USING AN RTE EXEC CALL.

SOME CONVENTIONS

- NUMERIC PARAMETERS ARE ASSUMED TO BE POSITIVE UNLESS PROCEEDED BY A MINUS (-) SIGN. PLUS (+) SIGN IS IGNORED.
- A NUMERIC PARAMETER FOLLOWED BY "B" IS OCTAL.
- TWO COMMAS (, ,) OR COLONS (: :) CAUSE A PARAMETER TO ASSUME ITS DEFAULT VALUE.
- LEADING BLANKS AND BLANKS ON EITHER SIDE OF A COMMA OR COLON ARE IGNORED.

ON, FMGR

TO SCHEDULE OPERATOR INTERFACE PROGRAM FMGR

*ON, FMGR [, *input* [, *log* [, *list* [, *severity code*]]]]

WHERE

input	IS THE LOGICAL UNIT NUMBER OF THE COMMAND INPUT DEVICE DEFAULT IS LU1 (SYSTEM TELEPRINTER)
log	IS THE LOGICAL UNIT NUMBER OF A DEVICE FOR LOGGING ERRORS. DEFAULT IS THE INPUT DEVICE. THE LOG DEVICE MUST BE TWO-WAY, THUS IF INPUT IS THE TAPE READER LOG DEFAULTS TO THE SYSTEM TELEPRINTER
list	IS THE LOGICAL UNIT NUMBER OF THE LIST DEVICE. DEFAULT IS LU6 (STANDARD LIST DEVICE)
severity code	IS THE ERROR MESSAGE SEVERITY CODE. DEFAULT IS Ø Ø – ECHO COMMANDS, PRINT ERRORS 1 – INHIBIT COMMAND ECHO 2 – INHIBIT ERROR MESSAGES UNLESS SEVERE ENOUGH TO REQUIRE OPERATOR ACTION.

LLu (LIST FILE CHANGE)

TO CHANGE THE CURRENT ASSIGNMENT OF THE LIST FILE

:LL, *namr*

WHERE

namr IS THE NAME OF A FILE, OR LOGICAL UNIT NUMBER

THE LIST FILE IS WHERE THE COMMANDS LI, CL, AND DL DIRECT THEIR OUTPUT.

LOglu (LOG DEVICE CHANGE)

TO CHANGE THE LOGICAL UNIT NUMBER OF THE SYSTEM LOG DEVICE.

:LO, *lu*

WHERE

lu IS THE LOGICAL NUMBER OF THE NEW LOG DEVICE

NOTE

lu MUST BE A TTY TYPE DEVICE (*i.e.* DVR ØØ) AND CANNOT BE A FILE NAME.

SeVerity code change

TO CHANGE THE SYSTEM LOG SEVERITY CODE TO A NEW NUMBER.

:SV, *number*

WHERE

number IS THE NEW SEVERITY CODE NUMBER

Ø – PRINT COMMANDS AND ERROR MESSAGES ON LOG DEVICE

1 – INHIBIT COMMAND PRINT-OUT ON LOG DEVICE BUT PRINT ERROR MESSAGES

2 – INHIBIT ERROR MESSAGES ON LOG DEVICE UNLESS SEVERE ENOUGH TO REQUIRE OPERATOR ACTION

- SV IS MOST USEFUL WHEN USED FROM WITHIN A COMPLETELY DEBUGGED TRANSFER FILE TO LIMIT PRINTOUT.

??

TO EXPAND THE LAST ERROR MESSAGE.

:?? [,*number*]

WHERE

number IS THE ERROR CODE.

IF *number* = BLANK – LAST ERROR CODE ISSUED IS EXPANDED.

IF *number* = XX – THE XX ERROR CODE IS EXPANDED

IF *number* = 99 – ALL ERROR CODE MESSAGES ARE PRINTED ON THE LIST FILE

EXit

TO TERMINATE THE FILE MANAGER (FMGR).

:EX

FMGR SCHEDULE

TO PROGRAMMATICALLY SCHEDULE FMGR

FORTRAN CALL

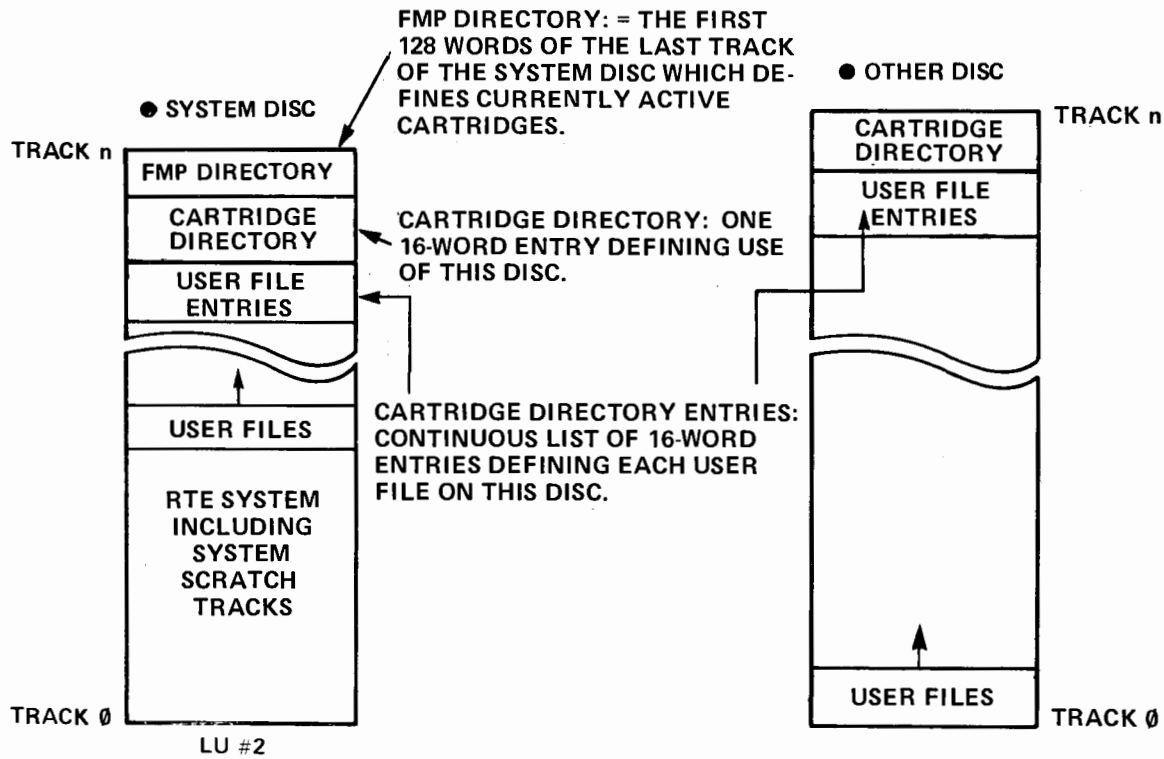
```
DIMENSION NAME (3)
    •
    •
    •
NAME (1) = 2HFM
NAME (2) = 2HER
NAME (3) = 2H
    •
    •
    •
CALL EXEC (ICODE,NAME,IP1,IP2,IP3,IP4,IP5)
```

ICODE = 23 (QUEUE SCHEDULE WITH WAIT)
24 (QUEUE SCHEDULE WITHOUT WAIT)

PARAMETERS IP1 THROUGH IP5 MAY BE DEFINED AS FOLLOWS:

IP1	*input device*		IP1	NAME OF FILE FROM WHICH
IP2	*log device*		IP2	FMGR IS TO READ ITS COMMANDS
IP3	*list device*	–OR–	IP3	WHEN IT IS TURNED ON
IP4	*severity code*		IP4	*severity code*
IP5	NOT USED		IP5	*list device*

FMP DISC ORGANIZATION



Mount Cartridge

TO NOTIFY THE FMP THAT A CARTRIDGE HAS BEEN MOUNTED AND IS AVAILABLE FOR USE.

:MC, *Lu* [, *Last track*]

FMP DIRECTORY

Ø	LU FIRST DISC
1	LAST TRACK FOR FMP
2	LABEL WORD
3	LOCK WORD
4	LU - SECOND DISC ETC. ⋮ Ø - END OF DISC SPECS (UP TO 31 DISCS)
124	Ø
125	SET UP (INITIALIZED) CODE WORD
126	MASTER SECURITY CODE
127	SPARE

WHER'

Lu	IS THE LOGICAL UNIT NUMBER OF THE CARTRIDGE BEING MOUNTED.
Last Track	IS LAST TRACK ON THE CARTRIDGE AVAILABLE TO THE FMP (NEW LAST TRACK OR AS REPORTED BY THE DC COMMAND) THE LAST TRACK IS THE DIRECTORY TRACK. FOR LU2 AND LU3, IF LAST TRACK IS NOT PROVIDED, FMP USES THE SYSTEM DEFINED LAST TRACK FOR THE SUBCHANNEL.

- MC PLACES THE ENTRY AT THE BOTTOM OF THE FMP DIRECTORY, HOWEVER, THE ORDER OF THE FMP DIRECTORY CAN BE CHANGED BY USING THE DC COMMAND.

A BRAND NEW VIRGIN CARTRIDGE MUST BE FORMATED PRIOR TO ITS USE BY THE FMP.

A CARTRIDGE MUST BE MOUNTED BEFORE BEING INITIALIZED BY THE IN COMMAND.

VIRGIN CARTRIDGE INITIALIZATION

During initial moving head system generation, the operator is given the option of formatting all sub-channels. The following procedure may be used to format new packs, or packs that may not have been mounted during the initial generation.

1. Load the moving head RTGEN using the BBL.
2. Load and configure a TTY SIO driver.
3. Start RTGEN at octal address 100; clear the switch register and push RUN.
4. Answer the cartridge channel question as in a normal generation.
5. a. Define the areas on the respective sub-channels (number of tracks and starting track number) which are to be initialized.

   b. Define a subchannel higher than any defined in "a" that is not mounted (i.e., either the unit is not present or it is unloaded). For example; if the system has one drive (i.e., subchannels 0 and 1), then define some tracks on subchannel 2. Then enter the /E.

CAUTION

Do not define any cartridge areas in step 5.a that have existing systems or data that is to be preserved on them. Steps 6 through 9 are important to prevent RTGEN from automatically formatting a cartridge area containing data.

6. Assign the dummy subchannel as the system sub-channel.
7. Assign the dummy subchannel as the scratch data.
8. Do not assign an "AUX DISC".
9. Set scratch origin to 0.
10. The number of 128 word sectors per track is 48 for the HP 7900/7901 and 24 for the HP 2870.
11. Answer the rest of the questions as per a normal generation.
12. When RTGEN asks:

    INITIALIZE SUBCHNL:
    X?

    answer YES for those channels to be initialized.
13. After the last subchannel is initialized, RTGEN will try to initialize the dummy subchannel (because it is the system subchannel) and will find it down. This causes message:

    READY DISC AND PRESS RUN

At this time the cartridges are formatted and the procedure is complete.

Dismount Cartridge

TO MAKE A CARTRIDGE UNAVAILABLE TO THE FMP, AND REMOVE ITS ENTRY FROM THE DISC DIRECTORY

:DC, *Label*

WHERE

label IS THE CARTRIDGE REFERENCE, POSITIVE FOR CR OR NEGATIVE FOR LOGICAL UNIT NUMBER.

- *THE DC COMMAND MUST BE ISSUED BEFORE A CARTRIDGE IS REMOVED.*
- THE LAST TRACK ON THE CARTRIDGE, AVAILABLE TO THE FMP, IS REPORTED ON THE LOG DEVICE. THIS SHOULD BE WRITTEN ON THE CARTRIDGE FOR USE WITH THE MC COMMAND.

The File Manager will not allow cartridges assigned to LU2 or LU3 to be physically removed with the DC Command. This is because LU2 and LU3 are both located in the RTE System track assignment table, and must stay mounted in order to properly configure the table when the system is initially started (booted). If the DC Command is issued to LU2 or LU3, the cartridge is locked and removed from the disc directory list. The cartridge is then immediately remounted at the bottom of the list. With this feature, the DC Command can be used to change the order of the cartridge directory list. For example, the directory list shown below is the order of cartridges immediately after the MC Command was used to mount the disc at LU14.

LU	LAST TRACK	CR	LOCK
02	0201	00002	
03	0201	00003	
14	0201	00055	

To place the peripheral disc (LU14) at the top of the directory list, issue the following commands (for this example only):

: DC,2

: DC,3

The File Manager will change the directory list as follows:

LU	LAST TRACK	CR	LOCK
14	0201	00055	
02	0201	00002	
03	0201	00003	

Application. The restriction on physically removing the cartridge assigned to LU3 can be circumvented if the following requirement and procedure are adhered to.

Requirement. All cartridges to be mounted at LU3 must be initialized to use the same first track. It is recommended that track 0 be selected as the first track to avoid the possibility of the loader or system placing a program in the area.

Procedure. Enter the following commands.

:DC, -3 (Insures all files are closed)

Remove the cartridge from the drive and insert the replacement.

:DC, -3 (Places new cartridge label in the disc directory)

The above procedure will work only if both cartridges have been initialized to use the same first track (recommended track 0). If the new cartridge has not been initialized FMGR will lock it. An attempt to initialize the new cartridge at this point will result in FMGR error 59 because the directory tracks are already assigned to D.RTR. This is solved as follows:

* RT, D.RTR (RTE command to release D.RTR tracks)

* ON, FMGR (FMGR will reassign D.RTR tracks on LU2, then terminate)

* ON, FMGR

: IN, *master security code*, -3 etc. (Initialize LU3)

This completes the procedure.

Application. Operator commands that use *namr* can take advantage of the default characteristic of the *namr* subparameter *label*. For example, when creating a file with the CR Command, and *label* is allowed to default to 0, the file is placed on the disc at the top of the directory list. Refer to the heading NAMR for more information on *label*.

INitialize

TO INITIALIZE CARTRIDGE PARAMETERS AND BUILD A CARTRIDGE DIRECTORY ENTRY, OR CHANGE THE MASTER SECURITY CODE.

:IN, [*master security code*] , *Label 1*, *Label 2*, *id* [, *1st trk* [,#*dir trks* [, #*sec/trk* [, *bad tracks*.]]]]

- OR -

:IN, *master security code* – – *new master security code* (CHANGE MASTER SECURITY CODE)

CARTRIDGE DIRECTORY ENTRY

0.	SIX-CHARACTER INFORMATIONAL PACK LABEL
1.	
2.	
3.	LABEL WORD
4.	FIRST AVAILABLE TRACK ON DISC
5.	NEXT AVAILABLE SECTOR
6.	#SEC/TR
7.	LAST AVAILABLE TRACK FOR FILES +1; I.E., LOWEST DIRECTORY TRACK
8.	–# TRACKS IN DIRECTORY
9.	NEXT AVAILABLE TRACK
10.	FIRST BAD TRACK (OR ZERO)
⋮	⋮
15.	SIXTH BAD TRACK (OR ZERO)

WHERE

master security code	IS THE TWO-CHARACTER FMP MASTER SECURITY CODE
Label 1	IS THE CARTRDIGE REFERENCE, POSITIVE FOR CR OR NEGATIVE FOR LOGICAL UNIT NUMBER. ALWAYS AN LU FOR CARTRIDGE NOT PREVIOUSLY INITIALIZED.
Label 2	IS THE NEW CARTRIDGE REFERENCE (CR) AND MUST BE > Ø .
id	IS THE CARTRIDGE IDENTIFICATION LABEL AND MUST HAVE THE CHARACTERISTICS OF A LEGAL FILE NAME. *id* IS PRINTED IN THE HEADING OF THE DIRECTORY LISTING.
1st trak	IS THE FIRST TRACK TO BE USED ON THE CARTRIDGE RELATIVE TO THE FIRST TRACK ASSIGNED TO RTE. NULL DEFAULTS TO TRACK Ø.
#dir trks	IS THE NUMBER OF DIRECTORY TRACKS (< 48). NULL DEFAULTS TO 1 TRACK
#sec/trk	IS THE NUMBER OF 64 WORD SECTORS PER TRACK. MUST BE SUPPLIED FOR CARTRIDGES NOT ON SAME CONTROLLER AS LU2 OR LU3.
bad tracks	IS THE BAD TRACK LIST. BAD TRACKS (ARE REPORTED WHEN FORMATING A VIRGIN CARTRIDGE) UP TO 6 TRACKS MAY BE ENTERED SEPARATED BY COMMAS

- *BE SURE TO REMEMBER THE NEW CODE. ONCE IT IS ENTERED, IT CANNOT BE OBTAINED WITH ANY FMGR COMMAND.*

EXAMPLES OF USE OF INITIALIZE

1. NEW CARTRIDGE INITIALIZATION

   :IN, SC, -14, 9600, CLASYS

2. CHANGE CARTRIDGE PARAMETERS

   :IN, SC, 9600, 9700, NEWSYS

   CR 9600 BECOMES CR 9700
   WITH THE NEW LABEL NEWSYS

3. PURGE ALL FILES AND COMPLETELY RE-INITIALIZE PARAMETERS

:IN, SC, -14, 9700, NEWSYS, 1	ATTEMPT TO CHANGE TRACK ASSIGNMENTS
FMGR Ø6Ø	FMGR WARNING MESSAGE
:YES	PURGES ALL FILES
:IN, SC, -14, 6500, CLEARS, Ø	NOW RE-INITIALIZE CARTRIDGE AND
:PK	RECAPTURE TRACK Ø.

- A CARTRIDGE ONLY HAS TO BE INITIALIZED THE FIRST TIME IT IS MOUNTED UNLESS ITS PARAMETERS ARE CHANGED.

Cartridge List

TO LIST THE ACTIVE CARTRIDGE LABELS AND THEIR STATUS

:CL

INFORMATION ON EACH MOUNTED CARTRIDGE IS LISTED AS FOLLOWS:

LU	LAST TRACK	CR	LOCK
Ø2	Ø2Ø2	ØØØØ2	
Ø9	Ø2Ø1	ØØ1ØØ	

- A CARTRIDGE MAY BE LOCKED BY FMGR DURING CERTAIN OPERATIONS SUCH AS PACKING THE DISC.

NAMR

- THE SYMBOLIC REPRESENTATION OF THE GENERAL FILE REFERENCE PARAMETER FOR THE FMGR COMMANDS
- IT IS CONSIDERED TO BE ONE PARAMETER, A FILE NAME, WITH UP TO FIVE SUBPARAMETERS SEPARATED BY COLONS (:)

namr = FILE NAME [: SECURITY CODE [: LABEL [: FILE TYPE [: FILE SIZE [: RECORD SIZE]]]]]

– OR –

namr = LOGICAL UNIT NUMBER (NOT A DISC)

FILE NAME

- A LEGAL FILE NAME IS SIX ASCII CHARACTERS MAXIMUM
- THE FIRST CHARACTER MUST NOT BE A BLANK OR A NUMBER
- ALL CHARACTERS MUST COME FROM THE SET BLANK THRU ← EXCLUDING + (PLUS) – (MINUS) : (COLON) , (COMMA)
- IMBEDDED BLANKS ARE NOT ALLOWED

EXAMPLES: FNAME
FNAME1

SECURITY CODE (SC)

- MAY BE ASCII, NUMERIC OR A COMBINATION OF BOTH
- AFTER CONVERSION BY FMGR THE MAGNITUDE SHOULD BE $\leq$32767
- IF SC = Ø FILE IS UNPROTECTED

  SC > Ø FILE IS WRITE BUT NOT READ PROTECTED

  SC < Ø FILE IS WRITE AND READ PROTECTED (MAY NOT BE OPENED WITHOUT SC)

IF SC IS ASCII ONLY THE FIRST TWO CHARACTERS ARE ACCEPTED AND THEN CONVERTED TO THE DECIMAL EQUIVALENT OF THE PACKED ASCII.

EXAMPLE:	SC	PACKED ASCII	DECIMAL EQUIVALENT
	AA	$Ø4Ø5Ø1_8$	$167Ø5_{10}$

CARTRIDGE REFERENCE (CR or LABEL)

- SPECIFIES A DISC PLATTER WITHIN THE FMP REALM ON WHICH A FILE IS TO BE STORED OR ALREADY EXISTS
- Ø<CR≤32767 – INDICATES A FMP "CARTRIDGE". THIS IS THE "NAME" OF THE CARTRIDGE.
- CR<Ø – USED TO INDICATE THE LOGICAL UNIT UPON WHICH A CARTRIDGE IS MOUNTED.
- CR=Ø – INDICATES THE FIRST AVAILABLE DISC THAT SATISFIES THE REQUEST IS TO BE USED

FILE TYPE

- A DECIMAL INTEGER IN THE RANGE Ø - 32767

CATEGORY	TYPE	DESCRIPTION
NON-DISK FILE	Ø	
FIXED LENGTHS RANDOM ACCESS NON-EXTENDABLE	1	128 WORD RECORD LENGTH
	2	USER SELECTED RECORD LENGTH
RANDOM LENGTH SEQUENTIAL ACCESS, AUTOMATIC EXTENTS	3	RANDOM RECORD LENGTH
	4	SOURCE PROGRAM
	5	RELOCATABLE PROGRAM
	6	RTE LOAD MODULE
	7	ABSOLUTE PROGRAM
	>7	USER DEFINED

- THE FILE TYPE NUMBER IS STORED IN THE DIRECTORY ENTRY FOR THE FILE. THE FILE MANAGER MUST REFERENCE THIS ENTRY FOR CERTAIN OPERATIONS, HOWEVER, FOR USER DEFINED FILE TYPES, THE FMP HAS NO INTEREST IN THIS ENTRY AND IT IS ASSUMED THE USER REFERS TO IT ACCORDING TO THE DICTATES OF HIS APPLICATION.

FILE SIZE

- FILE SIZE IS SPECIFIED IN "BLOCKS"

  1 BLOCK = 128 WORDS

- MINIMUM FILE SIZE IS ONE BLOCK

RECORD SIZE

- RECORD SIZE IS REQUIRED ONLY IF THE FILE TYPE IS 2

CReate file

TO CREATE A FILE NAME ON A CARTRIDGE IN WHICH DATA IS TO BE STORED LATER.

:CR, *namr*

CARTRIDGE DIRECTORY ENTRY

0.	SIX-CHARACTER FILE NAME OR	IF WORD 1 = Ø THEN THE END OF DIRECTORY ELSE
1.		IF WORD 1 = -1 THEN THE ENTRY WAS PURGED
2.		
3.	TYPE	
4.	STARTING TRACK	
5.	STARTING SECTOR/EXTENT	
6.	SECTORS IN FILE	
7.	IF TYPE = 2, (RECORD LENGTH)	
8.	SECURITY CODE	
9.		
10.		
11.		
12.	OPEN FLAGS (ADDRESS OF PROGRAM'S I.D. SEGMENT)	
13.		
14.		
15.		

WHERE

namr IS THE FILE NAME AND PARAMETERS

EXAMPLES:

```
:CR,MYFILE:-25:100:4:10
:LI,MYFILE:-25,D
MYFILE T=00004 IS ON CR00100 USING 00010 BLKS  R=0000

:CR,URFILE:::2:20:72
:LI,URFILE,D
URFILE T=00002 IS ON CR00002 USING 00020 BLKS R=0072

:CR,MTFILE:EJ:100:3:-1
:LI,MTFILE:EJ,D
MTFILE T=00003 IS ON CR00100 USING 01078 BLKS R=0000
```

- *NOTE: IF FILE SIZE IS -1 ALL AVAILABLE REMAINING SPACE ON THE CARTRIDGE IS ALLOCATED TO THE FILE.*

TYPE Ø FILE

- ALLOWS THE USER TO DECLARE AN I/O DEVICE A FILE
- STANDARD FILE COMMANDS AND CALLS CAN THEN BE DIRECTED TO NON-DISC DEVICES
- TYPE Ø FILES ARE CREATED AND PURGED BY THE OPERATOR INTERFACE, FMGR, AND RESIDE ON THE SYSTEM DISC

TYPE ZERO FILES ALLOW THE PROGRAMMER TO PERFORM DEVICE READS, WRITES AND CONTROL (REWIND, PUNCH LEADER, ETC.) USING STANDARD FMP LIBRARY ROUTINES.

DEVICE INDEPENDENCE MAY BE ACHIEVED BY USING FILE NAMES FOR DEVICES THEREBY ALLOWING DEVICE SWITCHING INDEPENDENT OF THE USER'S PROGRAM.

COOPERATING PROGRAMS MAY USE TYPE Ø FILES AS A MEANS OF CONTROLLING ACCESS TO A DEVICE THROUGH THE USE OF EXCLUSIVE FILE OPEN.

CReate type Ø file

TO CREATE TYPE Ø FILE WITH SPECIAL DIRECTORY ENTRIES FOR DEVICE CONTROL.

```
                  ,REad    [,BSpace   [,EOf      [,BInary   ]]]
:CR, namr, Lu     ,WRite   [,FSpace   [,LEader   [,ASciii   ]]]
                  ,BOth    [,BOth     [,PAge     [,numeric2 ]]]
                                      [,numeric1
```

CARTRIDGE DIRECTORY ENTRY

0., 1., 2.	SIX-CHARACTER FILE NAME OR { IF WORD 1 = -1 THEN THE ENTRY WAS PURGED
3.	Ø
4.	LOGICAL UNIT
5.	EOF CODE
6.	SPACING CODE
7.	READ/WRITE CODE
8.	SECURITY CODE
9., 10., 11., 12., 13., 14., 15.	OPEN FLAGS (ADDRESS OF PROGRAM'S I.D. SEGMENT)

WHERE

namr IS THE FILE NAME AND SECURITY CODE PARAMETER

Lu IS THE LOGICAL UNIT NUMBER OF THE DEVICE (NOT A DISC) TO BE CONTROLLED

numeric1 SAME AS I/O CONTROL FUNCTION CODE FOR THE RTE-II I/O CONTROL EXEC CALL. THIS IS A FIVE BIT FIELD EXPRESSED AS AN OCTAL CONSTANT OR DECIMAL CONSTANT:

REWIND = 05B

numeric2 SAME AS CONTROL WORD FUNCTION CODE BITS (X,A,K,V AND M) FOR RTE-II I/O READ/WRITE EXEC CALL. THIS IS A FIVE BIT FIELD EXPRESSED AS AN OCTAL OR DECIMAL CONSTANT.

- *THE REMAINING PARAMETERS SPECIFY LEGAL OPERATIONS , WHAT ACTION THE DEVICE IS TO TAKE WHEN AN END-OF-FILE IS ENCOUNTERED, AND DATA FORMAT.*

CReate type Ø file

THE *REad, WRite,* AND *BOth* PARAMETERS INDICATE THE LEGAL INPUT/OUTPUT.

REad INDICATES AN INPUT REQUEST IS LEGAL.

WRite INDICATES AN OUTPUT REQUEST IS LEGAL.

BOth INDICATES BOTH INPUT AND OUTPUT REQUESTS ARE LEGAL.

THE *BSpace, FSpace* AND *BOth* PARAMETERS INDICATE LEGAL SPACING. DEFAULT IS NO SPACING.

BSpace INDICATES BACKSPACING IS LEGAL.

FSpace INDICATES FORWARD SPACE IS LEGAL.

BOth INDICATES BOTH BACKSPACE AND FORWARD SPACING IS LEGAL.

THE *EOf, LEader,* AND *PAge* PARAMETERS ARE REQUIRED, AND SPECIFY THE CONTROL SUBFUNCTION FOR THE END-OF-FILE *WRITE* REQUEST. DEFAULT IS *LEader* IF THE DRIVER NUMBER IS 02_8. IF THE DRIVER NUMBER IS GREATER THAN 16_8, DEFAULT IS *Eof.* IF NONE OF THE ABOVE THEN DEFAULT IS PAGE.

EOf SPECIFIES AN END-OF-FILE MARK (MAGNETIC TAPE).

LEader SPECIFIES PAPER TAPE LEADER.

PAge SPECIFIES PAGE EJECT FOR LINE PRINTER, OR TWO LINE FEEDS FOR A TTY.

THE FOLLOWING PARAMETERS SPECIFY THE SUBFUNCTION PORTION OF THE INPUT/OUTPUT REQUEST. DEFAULT IS *AScii.*

BInary SPECIFIES BINARY DATA TRANSFER.

AScii SPECIFIES ASCII DATA TRANSFER.

Directory List

TO LIST THE CONTENTS OF FILE DIRECTORY OF ONE OR ALL OF THE MOUNTED CARTRIDGES.

:DL [,*Label* [,*master security code*]]

WHERE

Label	IS THE CARTRIDGE REFERENCE, POSITIVE FOR CR OR NEGATIVE FOR LOGICAL UNIT NUMBER.
master security code	IS THE TWO CHARACTER FMP MASTER SECURITY CODE DESIGNATED AT INITIALIZATION TIME

- *IF LABEL IS ABSENT (i.e. :DL OR :DL,, EJ) OR IT IS A Ø (i.e. :DL, Ø OR :DL,Ø, EJ) THEN ALL MOUNTED CARTRIDGE DIRECTORIES ARE LISTED.*

CARTRIDGE REFERENCE NUMBER

COMMAND

CARTRIDGE INFORMATION LABEL

NEXT AVAILABLE TRACK

NEXT AVAILABLE SECTOR

NO. OF SECTORS/TRACK

LAST FMP TRACK ON THIS DISC

NO. OF DIRECTORY TRACKS

:DL,2

```
CR-00002
   ILAB=SYSDIS NXTR=0133 NXSEC=074 #SEC/TR=096 LAST TR= 0202 #DR TR=02

 NAME      TYPE #BLKS/LU OPEN TO

 MT        00000     08
 REWND     00000     08  FMGR
 EOF       00004 00001
 TAPNUM    00002 00128  FMGR
```

FILE TYPE

FILE NAME

NUMBER OF BLOCKS USED BY FILE, OR A LOGICAL UNIT NUMBER IF FILE IS TYPE O.

FILE IS OPEN TO THIS PROGRAM. EXTENTS ARE INDICATED BY A PLUS (+) SIGN PRECEDING THE EXTENT NUMBER. EXCLUSIVE OPENS ARE INDICATED BY A MINUS (-) SIGN FOLLOWING THE PROGRAM NAME.

SHORT LIST

```
:DL,2,55

CR=00002
 ILAB=SYSDIS NXTR=0133 NXSEC=074  #SEC/TR=096 LAST TR=0202 #DR TR=02

NAME      TYPE   #BLKS/LU SCODE  TRACK SEC  OPEN TO

MT        00000      08   00000
REWND     00000      08   00000             FMGR
EOF       00004   00001   00000  0100  000
TAPNUM    00002   00128   12345  0100  002  FMGR
```

TRACK AND SECTOR ADDRESS

FILE SECURITY CODE

LONG LIST

Re Name

TO CHANGE A FILE NAME TO A NEW NAME

:RN, *namr*, *nuname*

WHERE

namr	IS THE EXISTING FILE NAME AND PARAMETERS
nuname	IS THE NEW FILE NAME UNIQUE TO THE CARTRIDGE

namr MUST NOT BE OPEN WHEN THIS COMMAND IS ISSUED. IF *namr* HAS A SECURITY CODE IT MUST BE SUPPLIED.

EXAMPLE: :RN, OLDFIL: EJ, NEWFIL

- IF *namr* EXISTS ON MORE THAN ONE CARTRIDGE, CR MUST BE SUPPLIED TO CHANGE THE DESIRED FILE NAME, OTHERWISE THE FIRST FILE ENCOUNTERED BY THAT NAME IS CHANGED.

PUrge

TO REMOVE A FILE AND ALL ITS EXTENTS FROM THE SYSTEM

:PU,*namr*

WHERE

namr IS THE NAME OF THE FILE TO BE PURGED, NOT A LOGICAL UNIT NO.

- *THIS COMMAND IS THE ONLY METHOD AVAILABLE TO REMOVE TYPE ØFILES.*
- *IF THE FILE BEING PURGED IS THE LAST FILE ON THE CARTRIDGE (EXCEPT TYPE 6 FILE) ALL ITS AREA IS RETURNED TO THE SYSTEM.*
- *SPACE ON A CARTRIDGE RESULTING FROM PURGING AN EMBEDDED FILE MAY BE RECOVERED BY PACKING OR PURGING ALL FILES AFTER IT.*

Pack

TO CLOSE UP GAPS IN BETWEEN FILES AND UTILIZE THE TRACKS LEFT FROM PURGED FILES.

:PK [,*label*]

WHERE

label IS THE CARTRIDGE REFERENCE, POSITIVE FOR CR OR NEGATIVE FOR LOGICAL UNIT NUMBER. IF *label* IS NOT SUPPLIED OR IS Ø THEN ALL CARTRIDGES ARE PACKED.

COpy files

TO COPY OR TRANSFER ALL FILES FROM ONE CARTRIDGE TO ANOTHER CARTRIDGE.

:CO, *Label 1*, *Label 2*

WHERE

Label 1 IS THE CARTRIDGE WHERE THE FILES ARE PRESENTLY RESIDING.

Label 2 IS THE NEW CARTRIDGE TO WHICH THE FILES ARE TO BE TRANSFERED.

BOTH *Label 1* AND *Label 2* CAN BE EITHER AN LU (-) OR CR (+).

- IF FILES WITH THE SAME NAME ARE LOCATED ON BOTH *Label 1* AND *Label 2* THE FILE ON *Label 1* IS NOT COPIED ONTO *Label 2*, BUT IS REPORTED WITH A FMGR -002 (DUPLICATE NAME ERROR.)

LIst

TO PRINT THE CONTENTS OF A FILE I THE SYSTEM LIST DEVICE.

:LI,*namr* [,*Source* / ,*Binary* / ,*Directory*]

WHERE

namr	*IS THE NAME OF THE FILE, OR A LOGICAL UNIT NUMBER.*
Source	*LIST IN ASCII SOURCE FORMAT*
Binary	*LIST IN BINARY FORMAT*
Directory	*LIST THE FILE HEADER ONLY*

LIst EXAMPLES

```
*ON,FMGR
:ST,1,MYFILE
TEST FILE FROM DISC.
I/O ERR ET EQT #02
:LL,1
```

```
:LI,MYFILE,S
MYFILE T=00003 IS ON CR00002 USING 00001 BLKS R=0000

0001  TEST FILE FROM DISC.
```

```
:LI,MYFILE,B
MYFILE T=00003 IS ON CR00002 USING 00001 BLKS R=0000

REC# 00001

052105 051524 020106 044514 042440 043122 047515 020104*TEST FILE FROMD
044523 041456                                          *ISC.
```

```
:LI,MYFILE,D
MYFILE T=00003 IS ON CR00002 USING 00001 BLKS R=0000
```

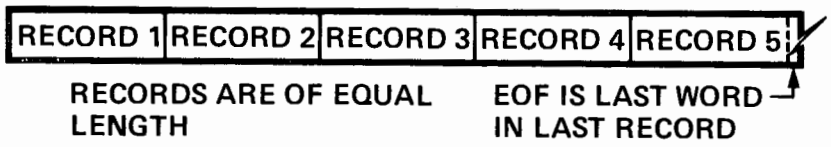
```
:LI,5,S
****** T=00000 IS ON LU 05

0001  TEST FILE FROM TAPE READER
```

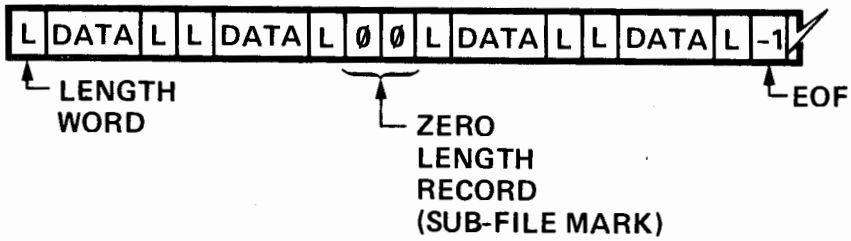
```
:EX
 $END FMGR
```

FMP FILE FORMATS

TYPE 1 & 2



TYPES 3 THROUGH 7 AND ABOVE (VARIABLE LENGTH RECORDS)



FOR THE PURPOSE OF THIS CLASS

SUBFILE MARK = ZERO LENGTH RECORD (DISC)

= TRUE EOF ON MAG TAPE

= LEADER BETWEEN MODULES ON PAPER TAPE

THE IDEA IS THAT PHYSICALLY INDEPENDENT MODULES STORED ON NON-DISC MEDIA CAN RETAIN THEIR IDENTITY WHEN STORED AND RETRIEVED FROM DISC.

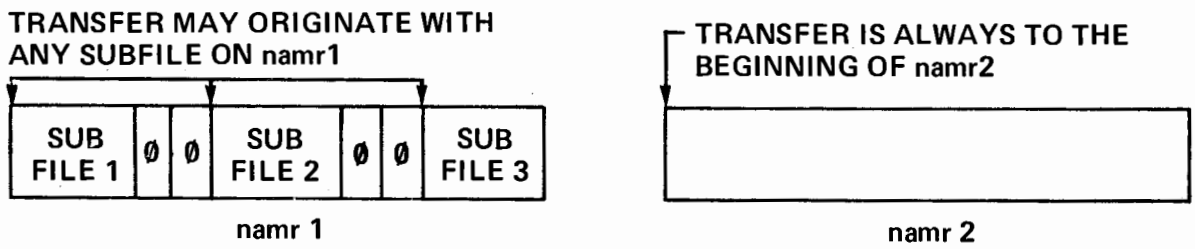
STore

TO TRANSFER OR STORE DATA FROM A FILE OR LOGICAL UNIT NUMBER, TO ANOTHER FILE OR LOGICAL UNIT NUMBER. A FILE IS CREATED BY THIS COMMAND.

:ST, *namr 1*, *namr 2* [, *record format*, EOF *control* [*origin*[,# *of subfiles*]]]

—OR—

:ST, *namr 1*, *namr 2* [, *record format* [,*origin* [, #*of subfiles*]]]
[, EOF *control*



WHERE

namr 1 & *namr 2*	IS THE NAME OF A FILE, OR LOGICAL UNIT NUMBER. DATA IS TRANSFERRED FROM *namr 1* TO *namr 2*.
record format	IS THE RECORD FORMAT EXPECTED FROM *namr 1*.
EOF control	SAVES EMBEDDED SUBFILE MARKS ON *namr 1* OR INHIBITS WRITING AN EOF AT THE END OF THE DATA ON *namr 2*.
subfile #	INDICATES THE RELATIVE SUB FILE FROM WHICH READING IS TO BEGIN ON *namr 1*
# subfiles	INDICATES THE NUMBER OF SUB FILES TO BE TRANSFERRED FROM *namr 1*

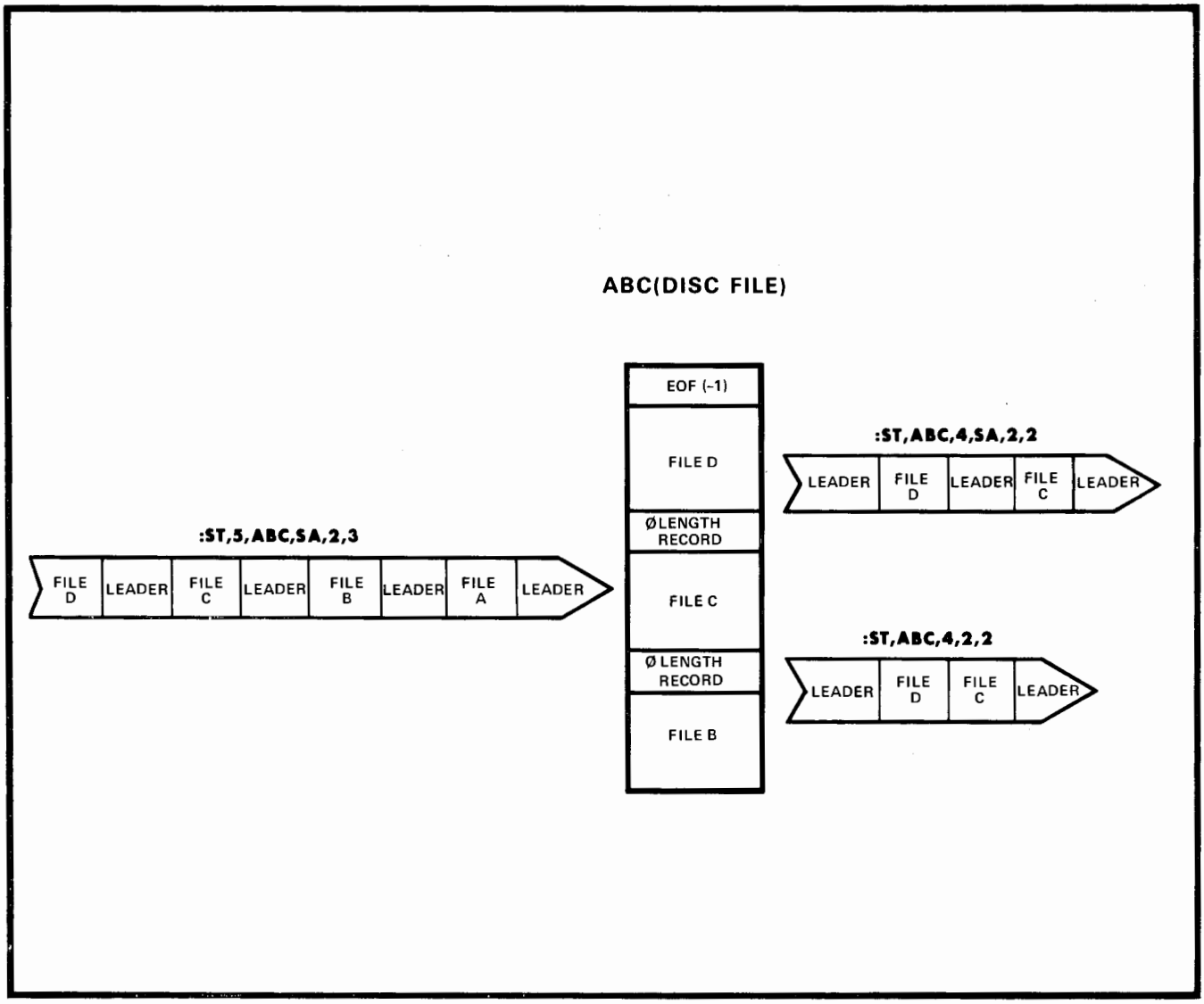
RECORD FORMAT IS A LITERAL AND MAY BE ONE OF THE FOLLOWING:

AScii	INDICATES THAT ASCII RECORDS ARE TO BE TRANSFERRED.
BReloc.	INDICATES THAT BINARY RELOCATABLE RECORDS ARE TO BE TRANSFERRED. A CHECKSUM IS DONE.
BNary	INDICATES THAT BINARY RECORDS ARE TO BE TRANSFERRED WITHOUT CHECKSUM.
BAbs.	INDICATES THAT BINARY ABSOLUTE RECORDS ARE TO BE TRANSFERRED. A CHECKSUM IS DONE.
MTape	INDICATES THAT MAGNETIC TAPE ASCII RECORDS ARE TO BE TRANSFERRED.
MS	INDICATES THAT MAGNETIC TAPE SIO (SYSTEM INPUT/OUTPUT) RECORDS ARE CREATED ON *namr2*. STANDARD RECORDS ARE EXPECTED ON *namr1*.
MSBR	INDICATES MAGNETIC TAPE SIO BINARY RELOCATABLE RECORDS (SAME AS MS + BR).
MSBA	INDICATES MAGNETIC TAPE SIO BINARY ABSOLUTE RECORDS (SAME AS MS + BA).

IF NOT SUPPLIED, DEFAULT IS DERIVED FROM THE FILE TYPE. IF FILE TYPE IS NOT SUPPLIED FOR *namr1*, FINAL DEFAULT IS ASCII.

EOF CONTROL IS A LITERAL AND MAY BE ONE OF THE FOLLOWING:

IHibit	INHIBITS WRITING AND END-OF-FILE MARK AFTER THE DATA. USEFUL ONLY WHEN *namr2* IS NOT A DISC FILE.
SAve	SAVES ANY SUBFILE MARKS IN *namr1* ON *namr2*. IN DISC FILES, SUBFILE MARKS ARE SAVED BY WRITING A ZERO LENGTH RECORD, AND INTERPRETED BY READING A ZERO LENGTH RECORD.
NOT SPECIFIED	EMBEDDED SUBFILE MARKS ARE NOT SAVED AND AN EOF IS ALWAYS WRITTEN AT THE END OF THE DATA.



NOTES:

WHEN INPUT IS FROM PAPER TAPE, THE FILE MANAGER WILL SUSPEND ITSELF AFTER READING A FILE (AT THE NEXT FILE'S LEADER). READING OF THE NEXT FILE IS ACCOMPLISHED BY TYPING:

GO, FMGR

A USEFUL TECHNIQUE FOR CREATING ASCII FILES FROM A CRT OR TTY IS AS FOLLOWS:

:ST, 1, FILEX, AS (CR)
<INPUT ASCII DATA> (CR)
:
<INPUT ASCII DATA> (CR)
CTRL-D (END OF FILE)

DUmp

TO TRANSFER OR DUMP DATA FROM A FILE OR LOGICAL UNIT NUMBER, TO ANOTHER FILE OR LOGICAL UNIT NUMBER. A FILE IS NOT CREATED BY THIS COMMAND.

:DU, *namr1*, *namr2* [,*record format*, EOF *control* [*destination* [#*subfiles*]]]

OR

:DU,*namr1*, *namr2* [,*record format* [*destination* [,#*subfiles*]]]
,EOF *control*

WHERE

namr1 & *namr2*	IS THE NAME OF A FILE, OR LOGICAL UNIT NUMBER. DATA IS TRANSFERED FROM *namr1* TO *namr2*.
record format	IS THE RECORD FORMAT OF *namr2*.
EOF *control*	SAVES EMBEDDED SUBFILE MARKS ON *namr2* OR INHIBITS WRITING AN EOF AT THE END OF DATA ON *namr2*.
file #	INDICATES THE RELATIVE FILE TO BE WRITTEN IN ON *namr2*, WHICH IS NORMALLY A NON-DISC DEVICE.
# files	INDICATES THE NUMBER OF SUBFILES TO BE TRANSFERRED FROM *namr1*

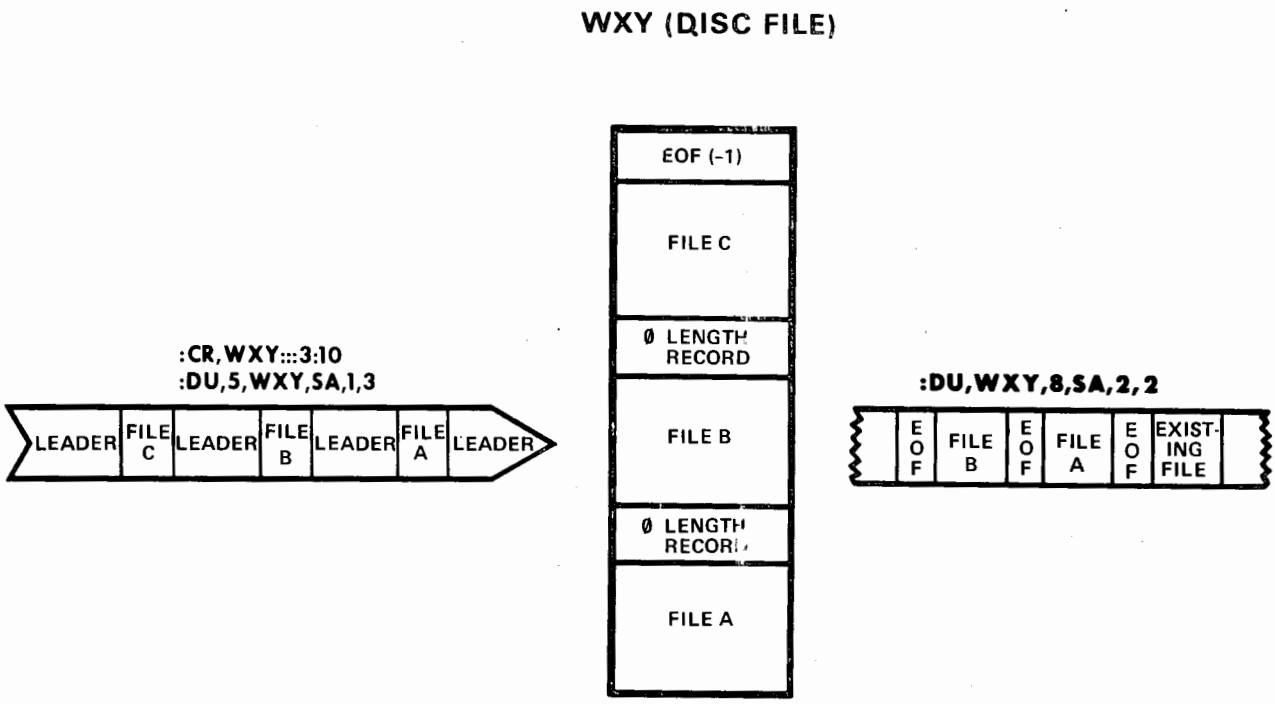
RECORD FORMAT IS A LITERAL AND MAY BE ONE OF THE FOLLOWING:

AScii	INDICATES THAT ASCII RECORDS ARE TO BE TRANSFERRED.
BReloc.	INDICATES THAT BINARY RELOCATABLE RECORDS ARE TO BE TRANSFERRED. A CHECKSUM IS DONE.
BNary	INDICATES THAT BINARY RECORDS ARE TO BE TRANSFERRED WITHOUT CHECKSUM.
BAbs.	INDICATES THAT BINARY ABSOLUTE RECORDS ARE TO BE TRANSFERRED. A CHECKSUM IS DONE.
MTape	INDICATES THAT MAGNETIC TAPE ASCII RECORDS ARE TO BE TRANSFERRED.
MS	INDICATES THAT MAGNETIC TAPE SIO (SYSTEM INPUT/OUTPUT) RECORDS ARE CREATED ON *namr2*. STANDARD RECORDS ARE EXPECTED ON *namr1*.
MSBR	INDICATES MAGNETIC TAPE SIO BINARY RELOCATABLE RECORDS (SAME AS MS + BR).
MSBA	INDICATES MAGNETIC TAPE SIO BINARY ABSOLUTE RECORDS (SAME AS MS + BA).

IF NOT SUPPLIED, DEFAULT IS DERIVED FROM THE FILE TYPE. IF FILE TYPE IS NOT SUPPLIED FOR *namr1*; FINAL DEFAULT IS ASCII.

EOF CONTROL IS A LITERAL AND MAY BE ONE OF THE FOLLOWING:

IHibit	INHIBITS WRITING AN END-OF-FILE MARK AFTER THE DATA. USEFUL ONLY WHEN *namr2* IS NOT A DISC FILE, THEREFORE CONCATONATION OF FILES IS USEFUL ONLY ON NON-DISC DEVICES.
SAve	SAVES ANY SUBFILE MARKS IN *namr1* ON *namr2*. IN DISC FILES, SUBFILE MARKS ARE SAVED BY WRITING A ZERO LENGTH RECORD, AND INTERPRETED BY READING A ZERO LENGTH RECORD
NOT SPECIFIED	EMBEDDED SUBFILE MARKS ARE NOT SAVED AND AN EOF IS ALWAYS WRITTEN AT THE END OF THE DATA.



NOTES:

WHEN INPUT IS FROM PAPER TAPE, THE FILE MANAGER WILL SUSPEND ITSELF AFTER READING A FILE (AT THE NEXT FILE'S LEADER). READING OF THE NEXT FILE MAY BE ACCOMPLISHED BY TYPING:

GO, FMGR

OPERATOR COMMANDS-
PROGRAM DEVELOPMENT

Move Relocatable

TO TRANSFER THE FILE AT *namr* TO THE LOAD-AND-GO AREA.

:MR, *namr*

THE RTE-II LG COMMAND IS IMPLEMENTED
IN FMGR AS:

:LG [, #OF TRACKS]

WHERE

namr IS THE NAME OF THE FILE OR LOGICAL UNIT NUMBER TO BE TRANSFERED.

- A CHECKSUM IS PERFORMED ON THE DATA TRANSFERED.
- *namr* MAY CONTAIN MORE THAN ONE RELOCATABLE MODULE. MODULES ARE TRANSFERED UNTIL AN EOF IS ENCOUNTERED.

Move Source

TO TRANSFER A FILE TO A SYSTEM LOGICAL SOURCE (LS) TRACK.

:MS, *namr* [, *progname* [, IH]]

THE RTE-II LS AND RT COMMANDS ARE IMPLEMENTED IN FMGR AS:

:LS [, LU, TRACK]

:RT, name of program owning tracks

WHERE

namr IS THE NAME OF THE FILE, OR LOGICAL UNIT NUMBER TO BE TRANSFERED.

progname IS THE NAME OF A PROGRAM TO WHICH THE LS TRACKS ARE TO BE ASSIGNED. IF NOT GIVEN THEY ARE ASSIGNED TO THE EDITOR. *Progname* RELEASES THE TRACK AFTER IT IS DONE WITH IT.

IH IS A LITERAL. IF SUPPLIED THE LS POINTER IS NOT SET TO POINT AT THE FILE JUST MOVED; AND THE RTE SYSTEM LS COMMAND IS REQUIRED.

EXAMPLE:

```
*ON,FMGR
:ST,1,MYFILE
MOVE SOURCE TEST FILE.
I/O ERR ET EQT #Ø2
:MS,MYFILE,FTN4,IH
FMGR Ø15
 LS LU 2 TRACK Ø37
:EX
 $END FMGR
```

IF FMGR WAS INTERNALLY SCHEDULED, THE SCHEDULING PROGRAM MAY OBTAIN THE LU AND TRACK NUMBER BY CALLING RMPAR.

SAve

TO SAVE THE LOGICAL SOURCE (LS) OR LOAD-AND-GO AREA OF THE SYSTEM IN A FILE

:SA [,LS / ,LG] , *namr*

WHERE

LS	INDICATES THE LOGICAL SOURCE AREA.
LG	INDICATES THE LOAD-AND-GO AREA.
namr	IS THE NAME OF A FILE (CREATED BY THIS COMMAND), OR A LOGICAL UNIT NUMBER.

RUN PROGRAM

TO SCHEDULE A PROGRAM FROM THE FMGR OPERATOR INTERFACE.

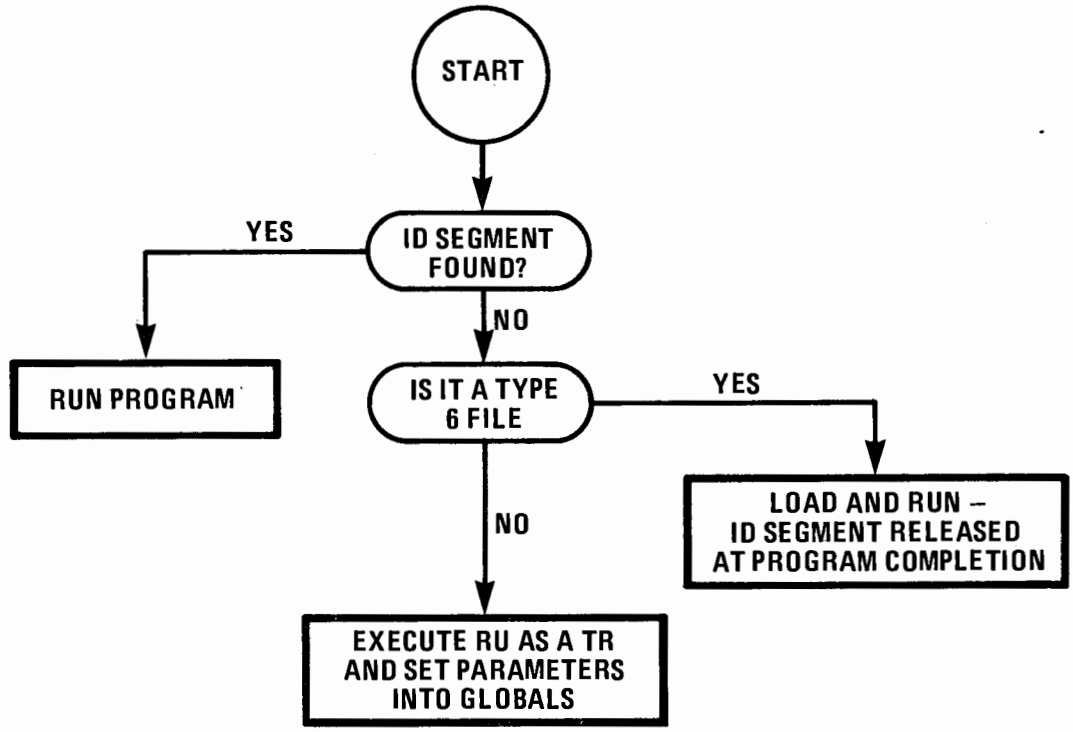
:RU, *name* [parameters]

WHERE

name MAY BE THE NAME OF A PROGRAM, TYPE 6 FILE OR TRANSFER FILE.

parameters ARE UP TO 5 OPTIONAL PARAMETERS THAT MAY BE PASSED TO NAME AS IN THE RTE-II "ON" COMMAND.

RU COMMAND INTERPRETATION



TERMINATE TEMPORARY BACKGROUND PROGRAM

:OF, *name*

NAME IS TERMINATED AND REMOVED FROM THE SYSTEM AS IN THE RTE-II OF, NAME, 8 COMMAND

Save Program

TO PLACE A DISC RESIDENT PROGRAM INTO A TYPE 6 FILE.

:SP, *namr*

WHERE

namr IS THE NAME OF A DISC RESIDENT PROGRAM. A LOGICAL UNIT NUMBERS IS ILLEGAL.

NOTE THAT THE PROGRAM NAME BECOMES THE BASIS FOR THE FILE NAME AS DESCRIBED BELOW.

- PROGRAMS WITH NAMES LESS THAN 5 CHARACTERS MUST BE SAVED USING ONLY THOSE CHARACTERS. ANY ATTEMPT TO ADD CHARACTERS COULD CREATE A CONFLICT WITH EXISTING PROGRAM NAMES.

  program = PROG
  :SP, PROG

- PROGRAMS WITH 5 CHARACTER NAMES MAY BE SAVED WITH AN APPENDED CHARACTER.

  program = PROGA
  :SP, PROGA1

- *SAVED PROGRAMS MAY BE ReNamed*

  :RN, PROG, PROGØ1

- *AND WHEN RESTORED TO THE SYSTEM, MAY BE ON'ED USING THE FIRST FIVE-CHARACTERS OF THE NEW NAME.*

  : RP, PROGØ1
  * ON, PROGØ

Restore Program

PURPOSE

TO RESTORE A PROGRAM (FILE), THAT WAS SAVED BY THE SP COMMAND, TO THE RTE SYSTEM.

FORMAT

:RP, *namr* (TO RESTORE A PROGRAM)

OR

:RP, *namr*, *program* (TO ASSIGN *program's* ID SEGMENT TO *namr*

OR

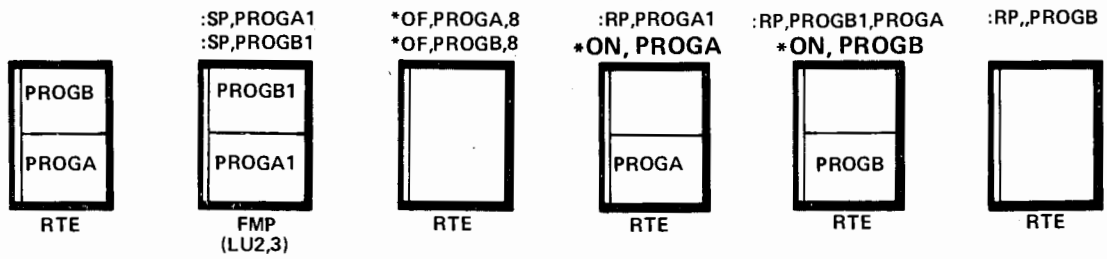
:RP, , *program* (TO RELEASE *programs* ID SEGMENT)

WHERE

namr IS THE NAME OF A TYPE 6 FILE ON LU2 OR LU3 WHICH WAS CREATED ON THE CURRENT SYSTEM USING THE SP COMMAND.

program IS A PROGRAM NAME.

Save Program - Restore Program



CONTROL NON-DISC DEVICES

:CN [, namr [, function [, subfunction]]] ISSUE I/O CONTROL TO DEVICE

namr	LU OR TYPE Ø FILE NAME – DEFAULTS TO LU8 (WHICH IS NORMALLY MAGNETIC TAPE)
function	RWind (DEFAULT ON DEVICE TYPES > 16) EOfile (DEFAULT ON DEVICE TYPES ≤ 16) TOform: LINE PRINTER SPACING FFile BFile FRecord BRecord LEader numeric = USED TO ISSUE CONTROL REQUEST NOT IN MNEMONIC LIST
subfunction	SPECIFIES CONTROL CODE FOR LINE PRINTER FORMATTING. DEFAULT GIVES TOP-OF-FORM ON LINE PRINTER AND 2 SPACES ON TTY/CRT.

FOR DVRØØ FUNCTION CODES 20B AND 21B ENABLE AND DISABLE RESPECTIVELY THE TERMINAL TO THE MULTI-TERMINAL-MONITOR.

TRansfer control

TO TRANSFER CONTROL OF FMGR TO A FILE OR LOGICAL UNIT NUMBER

:TR [,*namr* / ,-*integer* [,*parameter list*]]

WHERE

namr *IS THE NAME OF A FILE OR LOGICAL UNIT NUMBER.*

-integer *IS A NEGATIVE INTEGER THAT DENOTES A TRANSFER BACK THAT MANY STEPS.*

parameter list *FOR PASSING PARAMETERS TO GENERALIZED PROCEDURES*

- POINTS OF TRANSFER, *namr* AND CURRENT RECORD IF A DISC FILE, ARE RETAINED IN A STACK OF 10 MAXIMUM.
- TR (NO PARAMETER), CAUSES CONTROL TO BE TRANSFERRED BACK ONE ENTRY (IF ANY) IN THE STACK. IF THE STACK IS EMPTY, FMGR IS TERMINATED.
- TR, – *integer*, CAUSES CONTROL TO BE TRANSFERRED BACK IN THE STACK AS SPECIFIED BY – *integer*.

LABORATORY NO. 1 EXERCISE GUIDE

LESSON

FMP Operator Commands

OBJECTIVE

To provide the student an opportunity to experience the use of the FMP operator commands.

PROCEDURE

A. Carefully study each of the instructions listed below, then prepare a sequential list of the FMP operator commands required to accomplish each instruction. When you have completed the list go to a system and enter the commands as appropriate.

INSTRUCTIONS

1. Mount the removable cartridge.

2. Initialize the cartridge you have just mounted.

3. On the system console, obtain a list of all mounted cartridges.

4. Place the just-mounted cartridge at the top of the cartridge directory.

5. Create 4 files on the cartridge as follows:

File Name	Type	Size	Security Code
FILEA	Source	≈ 500 words	Your choice
FILEB	Source	≈ 500 words	Your choice
READER	Type Ø (for tape reader)		Your choice

6. Dump program tape TIME from file READER into file FILEA.

NOTE

Program tape TIME/PRIME contains a ASSEMBLY subroutine: TIME and FORTRAN main program: PRIME, separated by leader.

7. Dump program tape PRIME from READER into FILEB.

8. List the contents of files FILEA and FILEB on the standard list device.

9. Move the contents of FILEA and FILEB into separate system logical source tracks as required.

10. Using the system Load and Go option, perform the necessary steps to add program PRIME to the system as a background disc resident program.

11. Save program PRIME from the system in a file.

12. Remove program PRIME from the system.

13. Restore program PRIME to the system and run PRIME.

14. Obtain a directory listing, that includes file security codes, of all files that were created during this exercise and then purge the files.

15. Pack the cartridge.

16. Dismount the cartridge.

OPERATOR COMMANDS-
GENERALIZED PROCEDURES

PROCEDURES

A PROCEDURE IS A SET OF FMGR COMMANDS CONTRIVED TO PERFORM A PARTICULAR TASK.

BECAUSE THIS TASK IS OFT REPEATED, THE COMMANDS ARE STORED IN A TRANSFER FILE.

THEIR EXECUTION IS ACCOMPLISHED BY TRANSFERRING TO THE TRANSFER FILE USING THE TR COMMAND.

SIMPLE PROCEDURE TO COMPILE, LOAD AND RUN A PROGRAM

SYSTEM CONSOLE INPUT

```
*ON, FMGR
:TR, PROC1
:
```

PROCEDURE FILE PROC1

```
:LG, 1
:MS, FILEA
:RU, FTN4, 2, 99
:RU, LOADR, 99
:RU, PROGX
:OF, PROGX
:TR
```

RETURN OF CONTROL

- DRAWBACK: THE PROCEDURE WILL ONLY ACCEPT SOURCE FROM FILEA AND ONLY RUNS A PROGRAM NAMED PROGX.

LET'S GENERALIZE THE PROCEDURE BY LEAVING THE FILE NAME AND PROGRAM NAME UNDEFINED:

```
:LG, 1
:MS, <UNSPECIFIED>
:RU, FTN4, 2, 99
:RU, LOADR, 99
:RU, <UNSPECIFIED>
:OF, <UNSPECIFIED>
:TR
```

PROBLEMS:

1. WE MUST CODE SOME DUMMY OR UNDEFINED VARIABLE NAME INTO THE COMMANDS OR THEY WILL BE ILLEGAL.
2. WE MUST HAVE SOME WAY OF PASSING REAL NAMES TO THE PROCEDURE.

SOLUTIONS:

1. SYSTEM DEFINED VARIABLES THAT MAY BE REFERENCED IN PROCEDURE COMMANDS – WE CALL THESE *GLOBALS.*

2. A MEANS OF DEFINING THE GLOBALS BEFORE WE TURN CONTROL OVER TO THE PROCEDURE – WE DO THIS THROUGH THE *TR* OR *SET GLOBALS* COMMAND.

GLOBALS

VARIABLES THAT MAY BE SET, EXAMINED AND MANIPULATED BY JOB STREAM COMMANDS. THEY SERVE TO ALTER JOB COMMANDS SET UP AS A PROCEDURE THAT IS USED TO RUN A VARIETY OF JOBS WITH THE SAME COMMAND STREAM YET DIFFERING PARAMETERS WITHIN THOSE COMMANDS.

G-TYPE GLOBALS

1G – 9G VALUES SET BY TR, CA, AND SE COMMANDS

1ØG THE LOADER ALWAYS PASSES BACK IN 1ØG THE NAME OF THE PROGRAM IT JUST LOADED OR Ø IF THE PROGRAM WAS NOT LOADED

G-TYPE GLOBAL FORMAT

WORD	NULL	NUMERIC	ASCII
1	Ø	1	3
2	Ø	INTEGER	CHARACTERS 1, 2
3	Ø	Ø	CHARACTERS 3, 4
4	Ø	Ø	CHARACTERS 5, 6

WORD ONE IS USED TO INDICATE THE GLOBAL TYPE, I.E., NULL = TYPE Ø; NUMERIC = TYPE 1; ASCII = TYPE 3.

TRANSFER COMMAND USED TO PASS PARAMETERS AS GLOBALS

:TR [,namr / ,-integer [, p1, p1 . . . p9]]

- THE PARAMETERS p1 THRU p9 ARE USED TO SET GLOBALS 1G THRU 9G.
- POSITION OF PARAMETER IN LIST DETERMINES ORDINAL OF GLOBAL REFERENCED, I.E., P2 SETS 2G.
- IF ANY PARAMETER IS ABSENT, ITS VALUE REMAINS UNCHANGED

GENERALIZED PROCEDURE TO COMPILE, LOAD & RUN A PROGRAM.

SYSTEM CONSOLE INPUT	PROCEDURE FILE PROC1
*ON, FMGR	:LG, 1
:TR, PROC1, FILEA	:MS, 1G
:	:RU, FTN4, 2, 99
	:RU, LOADR, 99
	:RU, 1ØG
	:OF, 1ØG
	:TR

RETURN OF CONTROL

MESSAGE COMMANDS

:TE. message	MESSAGE IS SENT TO SYSTEM CONSOLE – USED FROM WITHIN A COMMAND FILE OR JOB INPUT STREAM. :TE APPEARS IN FRONT OF MESSAGE
:AN. message	SIMILAR TO *TE* EXCEPT *:AN* APPEARS IN FRONT OF MESSAGE.
:PA [,LU [,message]] ↓ :TR	SUSPENDS EXECUTION OF THE CURRENT JOB AND TRANSFERS CONTROL TO LU (DEFAULT IS LOG DEVICE). AN OPTIONAL MESSAGE MAY BE PRINTED ON THE LOG DEVICE. TO RESUME EXECUTION, ENTER TR ON DEVICE TO WHICH CONTROL WAS TRANSFERRED. *:PA* APPEARS IN FRONT OF MESSAGE.

- Only one line may be sent per command.

ASSUME USER MUST COMPILE A PROGRAM WHOSE SOURCE IS IN A FILE NAMED *RELO. RELO* CALLS TWO SUBROUTINES; ONE OF WHICH IS IN FILE SUBR; THE OTHER ON PAPER TAPE:

SYSTEM CONSOLE

```
*ON, FMGR
:TR, PROC2, RELO, SUBR
 PUT TAPE IN READER, TYPE "TR"
:TR
```

PROCEDURE FILE PROC2

```
:LG, 1
:MS, 1G
:RU, ASMB, 2, 99
:MR, 2G
:PA, PUT TAPE IN
  READER, TYPE "TR"
:MR, 5
:RU, LOADR, 99
:RU, 1ØG
:TR
```

CONDITIONAL BRANCHING IN JOB COMMAND STREAM

```
        EQ
:IF, p1, NE, p2 [,n]
        LT
        GT
        GE
        LE
```

COMPARES P1 AGAINST P2. IF THE CRITERIA IS MET, n COMMANDS WILL BE SKIPPED.

- IF n IS A NEGATIVE NUMBER OTHER THAN -1, THE JOB PROCESSOR SKIPS BACKWARD THAT NUMBER OF COMMANDS.
- n DEFAULTS TO 1.

- Do not attempt to skip beyond EOF or SOF.
- IF is illegal when executed from an interactive device.
- p1 and/or p2 may be ASCII, NULL or NUMERIC. The following relationships hold with mixed parameter types:

  NULL < NUMERIC < ASCII

TESTING 1ØG TO SEE IF PROGRAM WAS LOADED – SOURCE INPUT FROM PAPER TAPE (LU5)

SYSTEM CONSOLE	CONTENTS OF FORTLG:
:TR, FORTLG	:LG, 1 :RU, FTN4, 99 :RU, LOADR, 99 :IF, 1ØG, GT, Ø, 2 :AN, ***NO LOAD*** :AB :RU, 1ØG :TR

GLOBAL PARAMETER DEFINITION (SET COMMAND)

ALLOWS USER TO SET GLOBAL PARAMETERS 1G TO 9G ACCORDING TO VALUES IN P1 THRU P9.

:SE, P1, P2, P3 . . . . . P9

POSITION IN PARAMETER LIST DETERMINES ORDINAL OF GLOBAL REFERENCED, I.E., P2 SETS 2G

- IF ANY PARAMETER IS ABSENT ITS VALUE REMAINS UNCHANGED.
- IF ALL PARAMETERS ARE ABSENT, 1G THROUGH 9G ARE NULLED.
- GLOBALS CAN BE DISPLAYED ON THE SYSTEM CONSOLE WITH THE COMMAND:

  :DP, [PRAM1 [, PRAM 2 . . . .]]]

  EXAMPLE: :DP, 5G, 7G, 1P, 2P

WHERE 1P AND 2P ARE GLOBALS NOT YET DISCUSSED!

COMPUTE VALUE AND STORE IN GLOBAL

:CA, GLOBAL ID, P1, OP1, P2, OP2, P3, OP3, P4 . . . ETC.

RESULT STORED IN GLOBAL

COMPUTED FROM LEFT TO RIGHT

OPERATORS:

+	ADD
-	SUBTRACT
/	DIVIDE
*	MULTIPLY
O	OR
X	EXCLUSIVE OR
A	AND

EXAMPLES

:CA, 7, FTN4	THE GLOBAL 7G IS SET TO ASCII "FTN4".
:CA, 7, 2G	THE GLOBAL 7G IS SET TO THE VALUE IN 2G.
:CA, 5, 99	THE GLOBAL 5G IS SET TO INDICATE LGO.

GLOBALS 1P THROUGH 5P

- P-TYPE PARAMETERS REPRESENT INTEGER VALUES RETURNED TO FMGR BY A PROGRAM SCHEDULED BY THE FMGR.
- THE PROGRAM RETURNS THE PARAMETERS BY CALLING PRTN.
- ANOTHER PROGRAM REQUIRING THESE PARAMETERS MAY BE SCHEDULED IMMEDIATELY USING THE :RU COMMAND AS FOLLOWS:

  :RU, PROGX, 1P, 2P, 3P, 4P, 5P

EXAMPLE:

```
*ON, FMGR
:RU, PROG1
     <PROG1 CALLS PRTN TO RETURN VALUES TO FMGR IN
          1P THRU 5P>
:RU, PROG2, 1P, 2P, 3P, 4P, 5P
     <PROG2 PICKS UP THE PARAMETERS BY CALLING RMPAR>
```

OPERATOR COMMANDS-
BATCHED JOBS

JOB

A JOB IS A UNIT OF WORK TO BE DONE. USUALLY A PARTICULAR JOB IS ASSOCIATED WITH A PARTICULAR PROGRAMMER. ANY NUMBER OF PROGRAMS MAY BE EXECUTED WITHIN A JOB.

A JOB CAN ALSO BE DEFINED AS ALL THE WORK REQUESTED BETWEEN A :JO COMMAND AND AN :EO (END JOB) COMMAND.

JOB DEFINITION

:JO [, name [:hr:min:sec]

ALL WORK TO BE DONE BY PROGRAMMER *name*

:EO

INDICATES END OF JOB TO FMGR

WHERE

name	MAY BE UP TO 6 CHARACTERS
hr:min:sec	JOB EXECUTION TIME – DEFAULTS TO NO LIMIT

JOB EXAMPLE

```
*ON, FMGR
:JO, HAL: ØØ:1Ø:ØØ
:TR, FORTLG, INPUT
:EO
  <NEXT JOB>
```

```
:LG, 1
:MS, 1G
:RU, FTN, 2, 99
:RU, LOADR, 99
:TL, ØØ:Ø5:3Ø
:RU, 1ØG
:TR
```

TOTAL JOB TIME CANNOT EXCEED 1Ø MINUTES. RUNNING OF 1ØG CANNOT EXCEED 5 MINUTES.

LU TRANSFORM

:LU, REFERENCED lu, ACTUAL lu

CAN ONLY BE USED FROM WITHIN A JOB!

EXAMPLE 1. PROGRAM REFERENCES LUIO AS THE MAG TAPE, BUT MAG TAPE IS LU8.

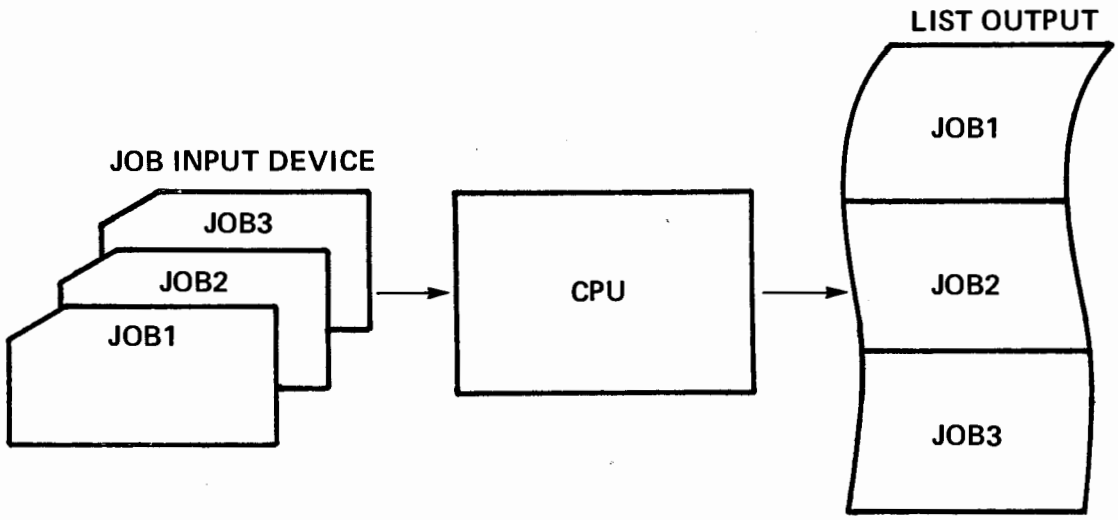
```
:JO, HAL
:LU, IO, 8    TRANSFORM REFERENCES TO LUIO TO LU8
   :
   :
 WRITE (1Ø, 35)
   :
   :
:EO
```

EXAMPLE 2. FOR THIS EXECUTION OF THE JOB, LIST IS TO GO TO LU7 (LP WITH SPECIAL FORMS) INSTEAD OF LU6

```
:JO, HAL
:LU, 6, 7
   :
   :
 WRITE (6, 4Ø)
   :
   :
:EO
```

BATCH

A BATCH IS SET OF JOBS THAT ARE TO BE RUN SEQUENTIALLY. THE SYSTEM MUST INITIALIZE CERTAIN RESOURCES AT THE COMPLETION OF EACH JOB.



END OF JOB PROCESSING-BATCH ONLY

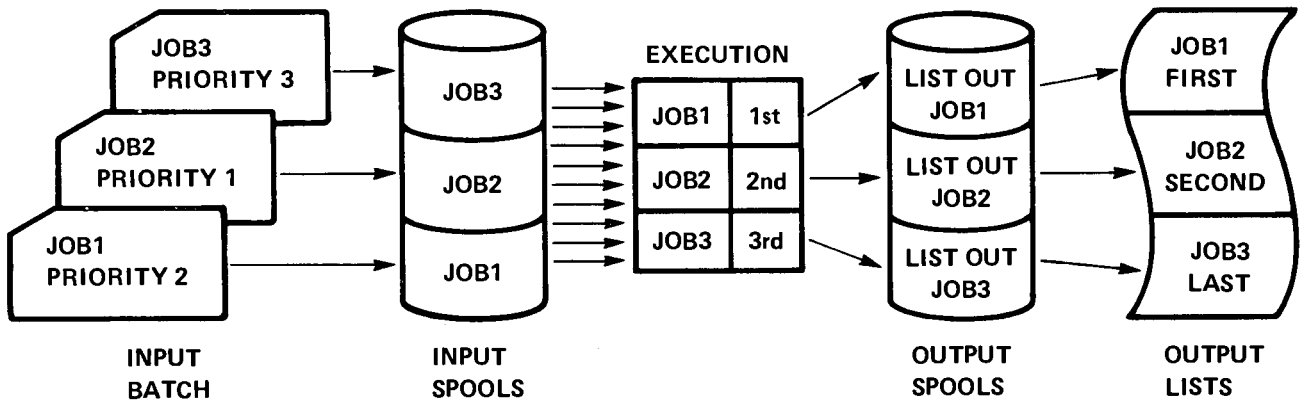
- RESETS LU MAP TO STANDARD DEVICES
- SENDS EOF TO LIST DEVICE
- CLEARS TRANSFER STACK
- CLEARS LOAD-AND-GO AND PARAMETERS 1ØG, IP THRU 5P*
- JOBFIL ENTRY STATUS CHANGED TO INDICATE "COMPLETE"
- INPUT UNIT CHECKED FOR NEXT COMMAND – IF INPUT IS FROM CARDS AND HOPPER IS EMPTY, FMP TERMINATES, OTHERWISE THE NEXT COMMAND IS READ. IF INPUT IS FROM A TRANSFER FILE, CONTROL RETURNS TO THE SYSTEM CONSOLE FOR THE NEXT COMMAND.

*GLOBALS 1G THROUGH 9G REMAIN DEFINED ACROSS JOB BOUNDARIES UNLESS SPECIFICALLY CHANGED BY A :TR OR :SE COMMAND.

OPERATOR COMMANDS-

SPOOLING

SPOOLING



- JOBS EXECUTE ON A PRIORITY BASIS EXCEPT FOR FIRST JOB, WHICH GOES INTO EXECUTION AS SOON AS SYSTEM IS AWARE OF ITS PRESENCE. LIKEWISE, ITS OUTPUT SPOOL FILE WILL BE DUMPED FIRST AS THE OUTSPOOL PROGRAM, SPOUT, DUMPS AS SOON AS OUTPUT IS AVAILABLE.

GASP

- GASP SCHEDULE REQUEST

  ON, GASP [, LU]

- TERMINATE GASP

  ↑EX

- ERROR MESSAGE INQUIRY

  ↑??

GASP INITIALIZATION

```
MAX NUMBER OF JOBS, JOB FILE DISC? 8,2
                                   -----
NUMBER OF SPOOL FILES (5 TO 80)? 24
                                 ---
SIZE OF SPOOL FILES (IN BLOCKS)? 24
                                 ---
NUMBER,LOCATION OF SPOOL FILES? 24
                                ---
NUMBER,LOCATION OF SPOOL FILES? E
                               ---
MAXIMUM NUMBER ACTIVE AND PENDING SPOOL FILES? 36
                                               ----
ENTER OUTSPOOL DESTINATION LU 6
                              ---
ENTER OUTSPOOL DESTINATION LU 7
                              ---
ENTER OUTSPOOL DESTINATION LU 4
                              ---
ENTER OUTSPOOL DESTINATION LU 1
                              ---
ENTER OUTSPOOL DESTINATION LU E
                              ---
```

SPOOL INITIATION

1. ON, JOB, lu

   LU OF DEVICE CONTAINING JOB

2. ON, JOB, 1
   :XEQ, file name, priority

   NAME OF FILE CONTAINING JOB. XEQ CAUSES FILE TO BECOME JOB INPUT SOURCE.

3. ON, JOB, FI, LE, XX, priority

   NAME OF FILE CONTAINING JOB – SAME AS XEQ COMMAND.

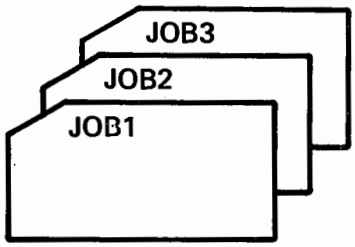
:JO COMMAND USED WITH SPOOLING

:JO, name:hr:min:sec, job priority, spool priority, NS

- SPOOLING OF JOB INPUT (LU5) AND LIST OUTPUT (LU6) IS *AUTOMATIC* UNDER JOB, HOWEVER OUTPUT SPOOLING CAN BE DEFEATED BY SPECIFYING NS ON THE :JO COMMAND.

RUNNING A JOB WITH SPOOLING

*ON,JOB,5



CARD READER
IS SYSTEM INPUT
DEVICE

```
*ON,JOB,FI,LE,XX
        OR
*ON,JOB,1
;:XEQ,FILEXX
<NEXT JOB OR CONTROL-D>
```

FILEXX =

```
:JO
:LG,1
:MS,FILEB
:RU,FTN4,2,99
:RU,LOADR,99
:RU,1ØG
:EO
```

```
*ON,JOB,1
;:JO
;:TR,FILEB,INPUT,OUTPUT
;:EO
<NEXT JOB OR CONTROL-D>
```

FILEB =

```
:LG,1
:MS,1G
:RU,FTN4,2,,2G,99
:RU,LOADR,99
:RU,1ØG
:TR
```

SPOOL FILE CHARACTERISTICS

- NAMES:

  SPOLØ1
  .
  .
  .
  SPOL8Ø

- ALL ARE SAME SIZE AS DEFINED AT SPOOL INITIALIZATION
- AUTOMATICALLY EXTENDED
- I/O TO SPOOL FILES IS DONE THROUGH STANDARD EXEC CALLS TO AN LU ASSOCIATED WITH THE SPOOL
- SPOOL LU'S AND THEIR CORRESPONDING EQT'S ARE DEFINED AT RTE - II SYSTEM GENERATION
- EACH SPOOL FILE RECORD IS PRECEDED BY TWO WORDS THAT SAVE THE I/O CONTROL INFORMATION SPECIFIED IN THE ORIGINAL I/O CALL

HOW SPOOL WORKS-
SIMPLIFIED

SPOOL PROGRAM MODULES

- JOB — JOB ENTRY MODULE – BUILDS JOBFIL ENTRY AND SPOOLS JOB TO DISC FILE
- SMP — SPOOL MONITOR PROGRAM – ASSIGNS AND MANAGES SPOOL FILES
- DVS43 — PERFORMS I/O TO AND FROM SPOOL FILES
- SPOUT — DUMPS SPOOL FILES TO ACTUAL OUTPUT DEVICES
- EXTND — CREATES EXTENTS FOR SPOOL FILES
- GASP — INITIALIZES JOBFIL AND SPOOL CONTROL FILE (SPLCON). PROVIDES OPERATOR COMMANDS TO EXAMINE AND MODIFY THE STATUS OF ONGOING JOBS AND SPOOLS.

SPOOL EQT'S AND LU'S

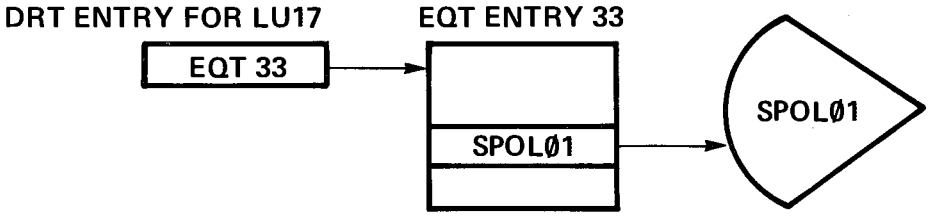
EQT 15? 31, DVS 43, X = 18	19 = EQT # ? 15
EQT 16? 32, DVS 43, X = 18	20 = EQT # ? 16
EQT 17? 33, DVS 43, X = 18	21 = EQT # ? 17
EQT 18? 34, DVS 43, X = 18	22 = EQT # ? 18
EQT 19? 35, DVS 43, X = 18	23 = EQT # ? 19
EQT 20? 36, DVS 43, X = 18	24 = EQT # ? 20

- The EQT entry must reference
- The X option for the EQT entry specifies an 18 word extension to the EQT.
- For each spool EQT, there must be a corresponding interrupt table entry, i.e.,

  31, EQT, 15

SPOOL I/O

1. SMP OBTAINS AN AVAILABLE SPOOL POOL FILE AND ASSIGNS AN AVAILABLE SPOOL LU TO IT:



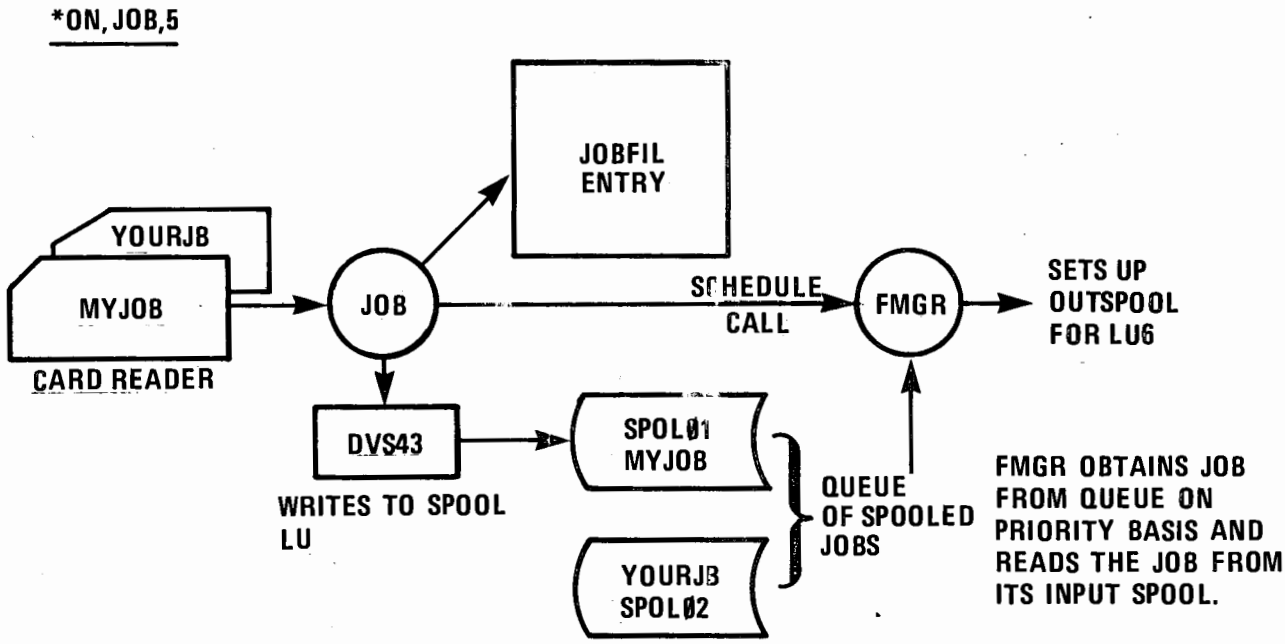
2. THE LU TRANSFORM TABLE IS SET SO THAT REFERENCES TO LU6 (STD. OUTPUT) BECOME REFERENCES TO LU17:

REFERENCED LU	ACTUAL LU
6	17

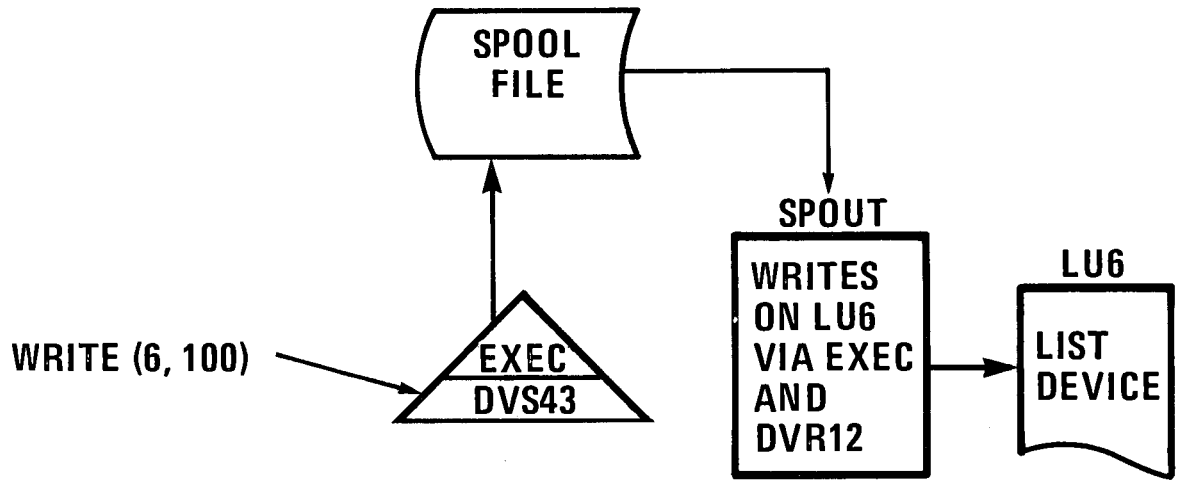
3. ALL SYSTEM OR PROGRAM REFERENCES TO LU6 NOW BECOME REFERENCES TO THE FILE SPOLØ1:

4. ALL I/O TO THE SPOOL FILES IS ACCOMPLISHED BY DVS43, WHICH ISSUES STANDARD EXEC CALLS TO THE SPOOL LU.

SIMPLIFIED JOB FLOW IN SPOOL SYSTEM



SIMPLIFIED LIFE OF AN OUTPUT SPOOL



THE JOB INPUT PROBLEM

THE FOLLOWING SPOOLED JOB WILL NOT WORK:

```
*ON,JOB,1
;:JO
;:LG,1
;:RU,FTN4,5,99                    INPUT EXPECTED FROM SYS. INPUT
;:RU,LOADR,99
;:RU,1ØG
;:EO
 <CNTRL-D>
```

WHY?

BECAUSE LU5 HAS BEEN REDEFINED AS A SPOOL FILE. THE ASSEMBLER WANTS TO READ ITS INPUT FROM LU5, WHICH IS NOW A SPOOL FILE. MEANWHILE, BACK AT THE CARD READER, THE INPUT SITS UNOTICED. ALSO, IF THE PROGRAM EXPECTS INPUT FROM THE SYSTEM INPUT DEVICE, IT WON'T GET IT BY ISSUING READS TO LU5.

ONLY IF THE SOURCE CODE AND DATA HAD BEEN ON THE SAME DEVICE AS THE JOB WOULD REFERENCES TO LU5 HAD WORKED AS THEN THE CODE AND DATA WOULD HAVE BEEN SPOOLED:

```
*ON,JOB,5 ---------------------> :JO
                                 :LG,1
                                 :RU,FTN4,5,99
                                 [ SOURCE ]
                                 :RU,LOADR,99
                                 :RU,1ØG
                                 [ DATA ]
                                 :EO
```

A SOLUTION:

```
ON,JOB,1
;:JO ------------------------- ASSIGNS LU1Ø = TO SYSTEM INPUT DEVICE
;:LU,10,5 -------------------- COMMAND REFERENCES LU1Ø
;:RU,FTN4,1Ø,99
;:RU,LOADR,99
;:RU,1ØG ------------------------ PROGRAM ISSUES READS FROM LU10
;:EO                              INSTEAD OF LU5
```

USER DEFINED SPOOL SETUPS

- USER CAN DESIGNATE HIS OWN NAMED FILES AS SPOOL FILES, WHICH PROVIDES –
- *GENERALIZED LU TO FILE EQUIVALENCE*

USER DEFINED SPOOL FILES

- THROUGH THE SPOOL SETUP COMMAND USER CAN DIRECT SMP TO ASSIGN SPOOL LU'S TO USER CREATED FILES.

JOB IS IN FILE *MINE*

LU5 = LU16 = MINE

OUTPUT IS TO GO TO FILE *LIST*

LU6 = LU17 = LIST

PUNCH OUT IS TO GO TO FILE BNRY

LU4 = LU15 = BNRY

GENERALIZED LU TO FILE EQUIVALENCE

- IN OTHER WORDS, YOU CAN REFERENCE ANY LEGITIMATE LU NUMBER (1-63) AND EQUATE THAT LU TO A PREVIOUSLY CREATED FILE OF YOUR CHOICE:

```
:JO
:LU,15,MYFILE. . . .
    .
    .
    .
PROGRAM TEST
    .
    .
    .
WRITE (15,101)
    .
    .
    .
END
    .
    .
    .
:EO
```

SPOOL SETUP

PURPOSE

TO SET UP SPOOL FILES FOR JOB INPUT AND/OR LIST OR PUNCH OUTPUT FROM/TO OTHER THAN STANDARD NON-DISK DEVICES.

:LU, lu, [namr] ,[attribute] ,[lu #] ,[spool priority]

- [spool priority] — DEFAULTS TO JOB PRIORITY
- [attribute] — SEE BELOW
- [lu #] — ACTUAL DESTINATION LU – USE OF THIS PARAMETER IMPLIES OUTSPOOLING
- [namr] — PREVIOUSLY CREATED FILE OR DEFAULTS TO SYSTEM MANAGED SPOOL FILE
- lu — LU USED TO REFERENCE SPOOL FILE

ATTRIBUTE:

HO	=	DO NOT GIVE FILE TO OUTSPOOLER UNTIL HOLD IS REMOVED OR SPOOL IS CLOSED
WR	=	WRITE ONLY AND NO HOLD
WH	=	WRITE ONLY AND HOLD
BO	=	READ/WRITE AND HOLD
BU	=	BUFFERED
SA	=	SAVE NAMR – DO NOT PURGE AFTER OUTSPOOLING
ST	=	STANDARD FILE – DOES NOT CONTAIN OUTSPOOL HEADER RECORDS
DEFAULT	=	READ ONLY, NO BUFFERING, NO-HOLD AND PURGE

The form

:LU, lu, namr

is used for input only spools.

CHANGE SPOOL SETUP BY LU COMMAND

```
         RWind = reset file pointer to first record
         PUrge = change SAve to PUrge
         SAve = change PUrge  to SAve
:CS, LU, PAss = remove HOld
         ENd = writes final EOF and terminates spool
         BUffer = change from NBuffer to BUffer
         NBuffer = change from BUffer to NBuffer
         NPass, [LU# [,PRIORITY]]
                 |          |
                 └─new LU   └─new priority
```

DEFAULT IS END.

END OF JOB PROCESSING-SPOOLING

- SENDS EOF TO OUTPUT SPOOLS
- CLEARS TRANSFER STACK
- CLEARS LOAD-AND-GO AND PARAMETERS 1ØG, 1P THRU 5P*
- CHANGES JOB'S STATUS TO "COMPLETE"
- JOB INPUT SPOOL FILE RETURNED TO SPOOL POOL
- LOOKS FOR NEXT JOB TO EXECUTE – SEARCHES JOBFIL FOR NEXT JOB, IF NONE, FMP TERMINATES

*GLOBALS 1G THROUGH 9G REMAIN DEFINED ACROSS JOB BOUNDARIES UNLESS CHANGED BY A :SE OR :TR COMMAND.

GASP (AGAIN)

- GASP SCHEDULE REQUEST

  ON, GASP[,LU]

- TERMINATE GASP

  ↑EX

- ERROR MESSAGE INQUIRY

  ↑??

SPOOL STATUS

- DISPLAY SPOOL STATUS

  ↑DS [,SPOOL LU]

LU	NAME	PRIORITY	JOB#	STATUS
6	SPOLØ1	0050	2	A

- CHANGE SPOOL STATUS

  ↑CS, SPOOL NAME, NEW PRIORITY / Hold spool / Release a previously held file

- PURGE SPOOL FROM OUTPUT QUEUE

  ↑KS, LU OF SPOOL / NAME OF SPOOL

  FILE CAN BE IN PROCESS OF BEING OUTSPOOLED UNLESS JOB IS ALSO ACTIVE, THEN JOB MUST BE ABORTED BEFORE KILLING SPOOL FILE

- RESTART AN OUTSPOOL FROM BEGINNING OF FILE

  ↑RS, SPOOL NAME (NEN LU)

  IF FILE IS NOT ACTIVE ONLY

STATUS=	A	FILE IS BEING OUTSPOOLED
	W	FILE IS WAITING FOR DEVICE
	H	FILE IS BEING HELD BY OPERATOR
	AH	FILE IS ACTIVE AND IN HOLD STATUS
	- -	FILE IS NOT AN OUTSPOOL FILE – LU WILL ALSO BE INDICATED AS "- -"

- Illegal to change the priority of an active file.
- Do not hold an active file - ties up device.

JOB STATUS

- ABORT JOB BEFORE IT RUNS

  ↑AB, JOB NUMBER—ONLY FOR JOBS IN RH OR R STATUS.

- DISPLAY JOB STATUS

  ↑DJ [JOB NUMBER / JOB NAME]

##	NAME	STATUS	SPOOLS
1	JOB4	S PPPP NN	1
2	JOBB	S PPPP NN	2
	JOB ORIGIN	PRIORITY	CURRENT STATUS

- CHANGE JOB STATUS

  ↑CJ, JOB NUMBER, NEW PRIORITY NUMBER / Hold job / Release previously held job

JOB STATUS:

I	JOB IS ENTERING SYSTEM
RH	READY TO RUN BUT IN HOLD
R	READY TO RUN
A	ACTIVE
CS	COMPLETED AND WAITING FOR OUTSPOOLING

It is illegal to alter the status of a job which is active or which is completed and waiting for outspooling to finish.

JOB ORIGIN:

S	INDICATES JOB IS FROM A FILE
D	INDICATES JOB IS DIRECT FROM AN LU

DEACTIVATE/ACTIVATE SPOOL SYSTEM

- SHUTDOWN SPOOL SYSTEM

  ↑SD [B – HOLD ALL PENDING JOBS / ,S – HOLD ALL PENDING SPOOLS]

  ACTIVE JOBS AND SPOOLS WILL RUN TO COMPLETION

- STARTUP SPOOL SYSTEM

  ↑SU [B – START UP HELD JOBS / ,S – START UP HELD SPOOLS]

  IF JOB OR SPOOL HAS PENDING HOLD FROM A PREVIOUS CJ OR CS COMMAND, THAT HOLD IS STILL IN EFFECT

- DEALLOCATE SPOOL FILES

  ↑DA

  THIS DEALLOCATES ALL SPOOL FILE DEFINED AT INITIALIZATION INCLUDING SPLCON AND JOBFIL

  EXAMPLE:

```
↑DA
KILLING SPOOL?
YES
SPOOLING DEAD
END GASP
```

LABORATORY NO. 2 EXERCISE GUIDE

LESSON

BATCH/SPOOL COMMANDS

OBJECTIVE

To provide the student an opportunity to use the BATCH/SPOOL MONITOR.

PROCEDURE

Refering to Steps 5 through 16 in Exercise No. 1, convert your solution to the use of Batching. Program names, program parameters, file names, file sizes, file types and file security codes should be passed as globals. This may require the use of multiple command files.

OPERATOR CONTROL-
OF SPOOL SYSTEM

FMP PROGRAM CALLS

USER PROGRAM CALLS

USER PROGRAMS COMMUNICATE WITH THE FMP THROUGH A SET OF TYPE 6 AND 7 UTILITY SUBROUTINES; THE FMP LIBRARY.

ALL DIRECTLY CALLABLE SUBROUTINES ARE TYPE 7 AND THEREFORE ARE APPENDED TO PROGRAMS CALLING THEM.

DATA CONTROL BLOCK (DCB)

A PROGRAM USING THE FMP MUST DECLARE A SCRATCH AREA (DCB) FOR EACH FILE OPEN TO IT.

IDCB BSS 144 + n

OR

DIMENSION IDCB (144 + n)

WHERE n = A MULTIPLE OF 128

THE DCB IS AN INTERFACE BETWEEN THE USER'S PROGRAM AND THE FMP. IT CONTAINS THE PACKING BUFFER AND CURRENT STATUS (16 WORDS) ON THE FILE IT IS OPEN TO.

- *ONE DCB WITHIN A PROGRAM MAY BE SHARED AMONG SEVERAL FILES IF ONLY ONE FILE NEED BE OPEN TO THE PROGRAM AT A TIME.*

DCB SIZE

- SPECIFIED IN IDCB PARAMETER OF CREATE OR OPEN CALLS OR DEFAULTS TO 144 WORDS.
- CONSISTS OF A 16-WORD CONTROL AREA AND 128 + n WORD BUFFER AREA.
- BUFFER SIZE ACTUALLY USED IS DETERMINED BY THE FOLLOWING:
  a. BUFFER SIZE USED IS ALWAYS A MULTIPLE OF 128.
  b. BUFFER SIZE USED IS NEVER GREATER THAN DECLARED BUFFER SIZE.
  c. BUFFER SIZE USED IS THAT MULTIPLE OF 128 THAT COMES CLOSEST TO, BUT NEVER EXCEEDS FILE SIZE.

FMP ERRORS

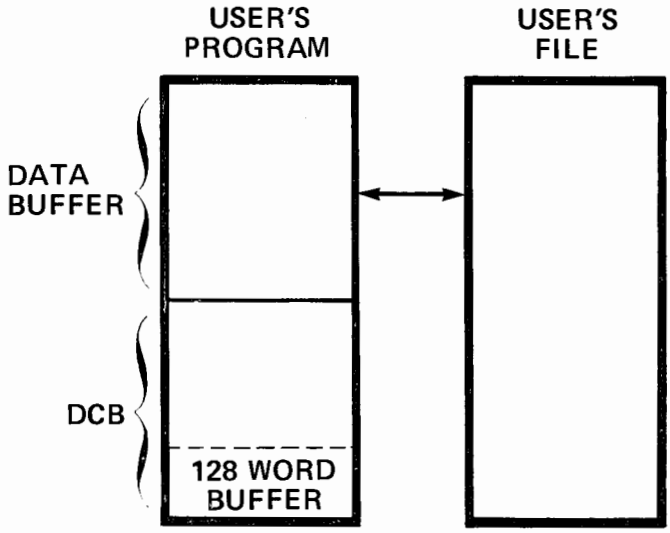
EACH LIBRARY SUBROUTINE RETURNS ERROR INFORMATION (IF AN ERROR EXISTS) THEREFORE THE USERS PROGRAM SHOULD BE CAPABLE OF RESPONDING TO THIS INFORMATION.

```
•
•
•
CALL PURGE (IDCB, IERR, NAME)
CALL IER (IERR)
```

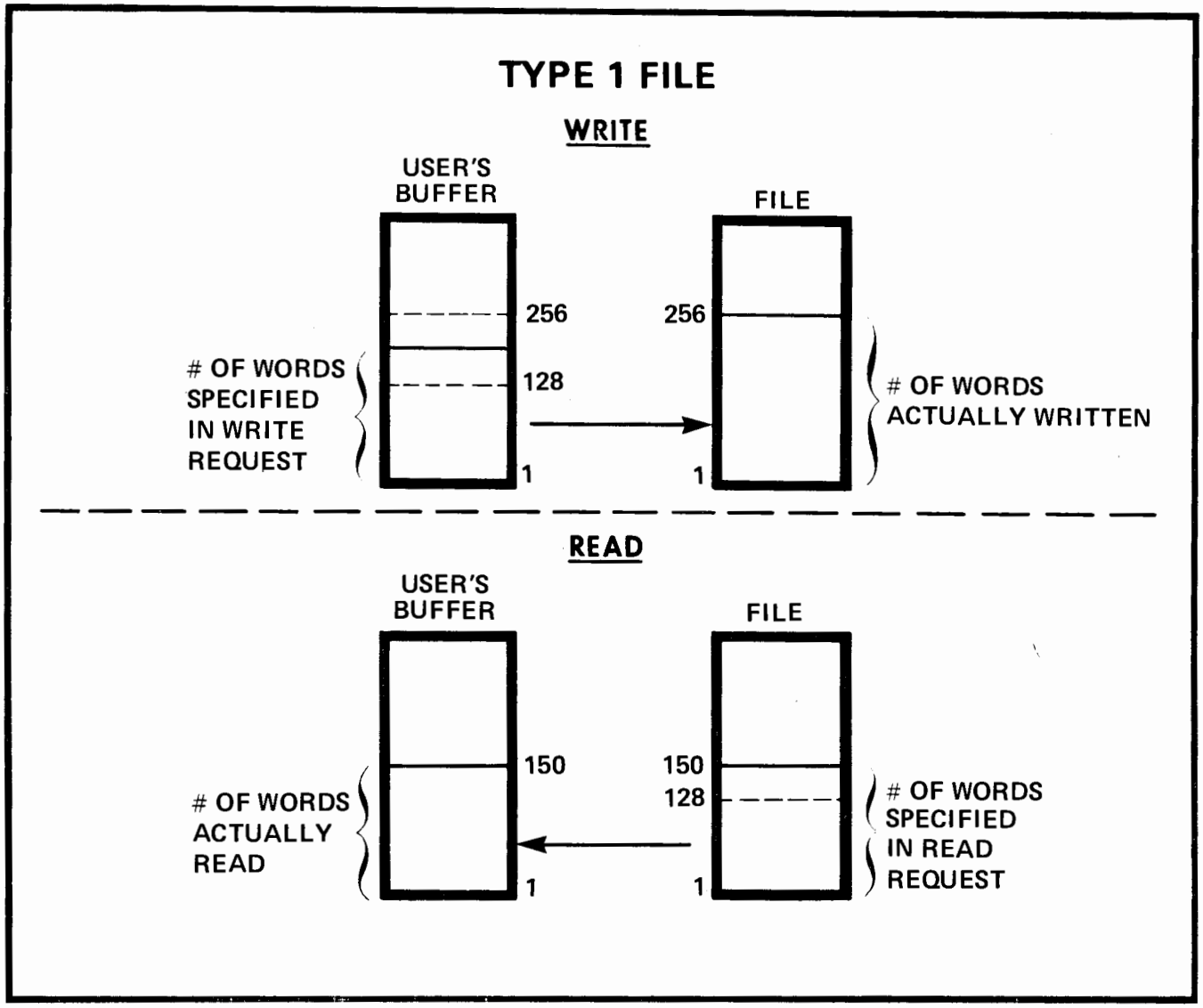
TYPE 1 FILE

- RECORDS HAVE A FIXED LENGTH OF 128 WORDS AND MAY BE ACCESSED IN RANDOM ORDER
- TRANSFERS MAY BE ANY NUMBER OF WORDS IN LENGTH
  - FOR WRITES THE NUMBER OF WORDS ARE ROUNDED UP TO MULTIPLES OF 128 (USING WORDS FROM THE USER'S BUFFER)
  - FOR READS THE EXACT NUMBER OF WORDS ARE TRANSFERRED
- TYPE 1 FILES HAVE THE FASTEST TRANSFER RATE

TYPE 1 FILES HAVE THE FASTEST TRANSFER RATE BECAUSE TRANSFERS ARE DIRECTLY TO OR FROM THE USER'S BUFFER.



DATA IS WRITTEN IN THE FILE AS PRESENTED AND THE END-OF-FILE IS THE LAST WORD OF THE LAST BLOCK IN THE FILE.

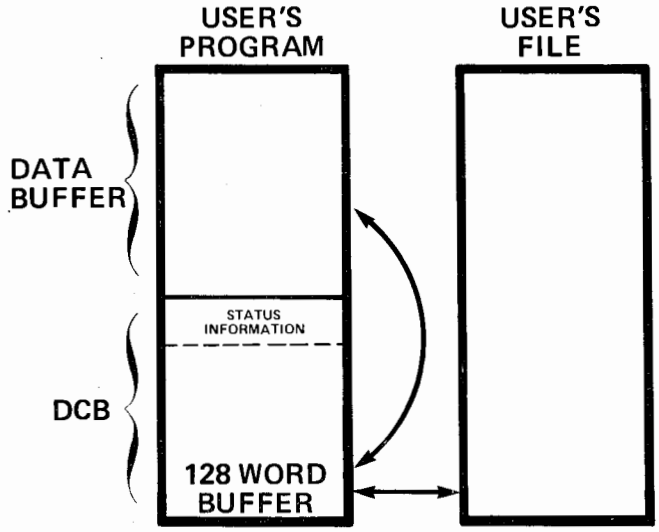


- IN THE READ EXAMPLE ABOVE, THE DISK DRIVER ACTUALLY MUST READ 256 WORDS FROM THE FILE (2 SECTORS), BUT ONLY 150 ARE PASSED FROM THE DRIVER TO THE USER'S PROGRAM.
- ALL FILES EXCEPT TYPE Ø FILES MAY BE READ AS TYPE 1 FILES, I.E., DIRECT TRANSFER THROUGH THE USERS BUFFER.

TYPE 2 FILE

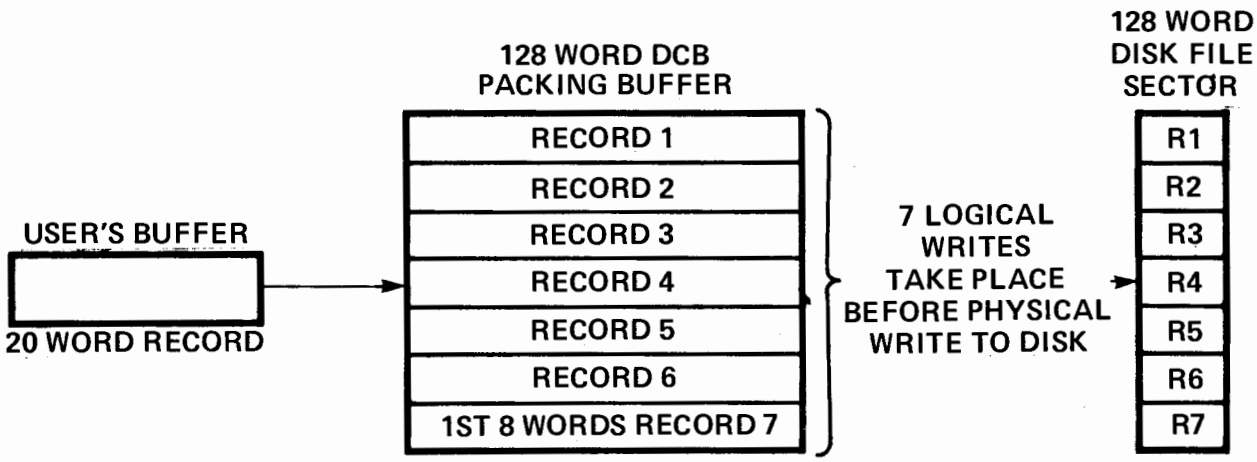
- RECORDS ARE FIXED IN LENGTH, AS DEFINED BY THE USER, AND THEY MAY BE ACCESSED IN RANDOM ORDER
- MAXIMUM RECORD SIZE IS 32767 WORDS
- EACH TRANSFER IS ONE AND ONLY ONE RECORD LONG
- TRANSFER RATE IS SLOWER THAN TYPE 1 FILES

TYPE 2 AND ABOVE FILES HAVE A SLOWER TRANSFER RATE THAN TYPE 1 BECAUSE DATA MUST GO THROUGH A PACKING BUFFER.



DATA IS WRITTEN IN THE FILE AS PRESENTED AND THE END-OF-FILE IS THE LAST WORD OF THE LAST BLOCK IN THE FILE.

LOGICAL VS. PHYSICAL WRITES (SEQUENTIAL)



- ESSENTIALLY THE REVERSE TAKES PLACE ON READS. IF A USER'S PROGRAM MAKES A READ REQUEST AND THE RECORDS ARE NOT IN CORE, THE DCB IS FILLED BY A PHYSICAL READ FROM THE DISC AND LOGICAL RECORDS ARE THEN PASSED AS REQUIRED TO THE PROGRAM BUFFER.
- RANDOM READ/WRITE IS ALSO THE SAME EXCEPT THAT PHYSICAL ACCESSES MAY BE MORE FREQUENT AS SUCCESSIVE REFERENCES TO RECORDS ARE LESS LIKELY TO BE SATISFIED BY THE BLOCK CURRENTLY IN CORE. ALSO, RANDOM UPDATING OF RECORDS REQUIRES THAT BLOCK BE READ BEFORE MODIFICATION.

TYPE 3 AND ABOVE FILES

- RECORDS ARE RANDOM IN LENGTH AND ARE ACCESSED SEQUENTIALLY
- EACH TRANSFER IS ONE AND ONLY ONE RECORD LONG
- TYPE 3 AND ABOVE FILES WILL CONTAIN DATA, SOURCE, RELOCATABLE CODE, ETC.

FORMAT

L	DATA	L	L	DATA	L	Ø	Ø	L	DATA	L	L	DATA	L	-1

EXTENDABLE FILES

- TYPE 3 FILES AND ABOVE ARE AUTOMATICALLY EXTENDED WHEN A WRITE REQUEST LENGTH EXCEEDS SPACE AVAILABLE IN THE FILE
- EXTENTS ARE THE SAME SIZE AS THE BASE FILE, ARE ON THE SAME PLATTER, AND MAY NUMBER $\leqslant$ 256
- THE USER NEED NOT BE AWARE EXTENTS HAVE BEEN CREATED

- IF FILE ACCESS SPEED IS AN IMPORTANT CONSIDERATION, EXTENSIONS MAY BE AVOIDED BY DECLARING SUFFICIENT ROOM IN THE FILE TO ACCOMMODATE ALL THE DATA TO BE STORED IN IT.

CREAT

TO CREATE A FILE NAME ON A CARTRIDGE IN WHICH DATA IS TO BE STORED.

FORTRAN CALL

```
DIMENSION IDCB (144 + n), NAME (3), ISIZE (2)
                    •
                    •
                    •
CALL CREAT (IDCB, IERR, NAME, ISIZE, ITYPE, ISECU, ICR, IDCBS)
```

OPTIONAL: ISECU, ICR, IDCBS

n	EXTENDS DCB, MUST BE A MULTIPLE OF 128
IDCB	DATA CONTROL BLOCK
IERR	ERROR CODE OR NUMBER OF BLOCKS (X2) IN FILE
NAME	FILE NAME
ISIZE	2 WORD ARRAY. WORD 1 = REQUESTED FILE SIZE IN BLOCKS. IF NEGATIVE USE REST OF DISC. WORD 2 = RECORD LENGTH IF TYPE 2 FILE.
ITYPE	FILE TYPE NUMBER. **TYPE Ø FILE IS ILLEGAL.**
ISECU	SECURITY CODE
ICR	CARTRIDGE REFERENCE, POSITIVE FOR CR OR NEGATIVE FOR LOGICAL UNIT NUMBER
IDCBS	SIZE OF DCB BUFFER

- **IF IDCB WAS IN USE BY ANOTHER FILE, IT IS CLOSED, THEN OPENED EXCLUSIVELY TO THE CALLER ON SUCCESSFUL COMPLETION OF THE CALL.**

WRITF

THIS ROUTINE WRITES A RECORD ON THE FILE CURRENTLY OPEN TO THE DCB.

FORTRAN CALL

```
DIMENSION IDCB (144+n), IBUF (IL)
      •
      •
      •
CALL WRITF (IDCB, IERR, IBUF, IL, NUM)
```

OPTIONAL: IL, NUM

IL = WRITE REQUEST LENGTH
NUM = USED FOR RANDOM ACCESS POSITIONING

WRITE REQUEST FILE LENGTH (IL) vs. FILE TYPE

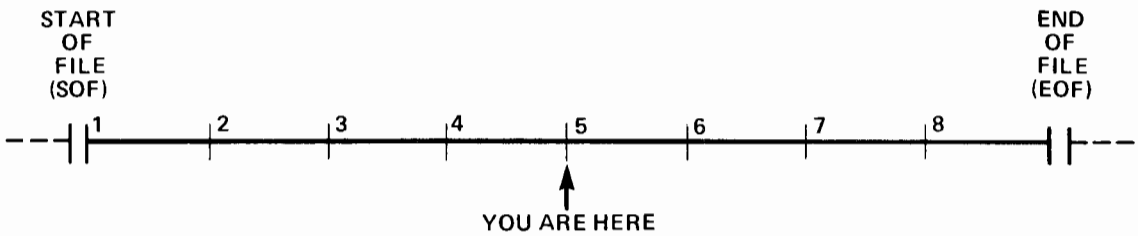
FILE TYPE \ REQUEST LENGTH	IL > 0	IL = 0	IL = -1
0	THE ROUTINE WILL WRITE EXACTLY IL WORDS.	A ZERO LENGTH RECORD IS WRITTEN.	AN END-OF-FILE IS WRITTEN.
1	DATA IS WRITTEN IN 128 WORD BLOCKS. FOR EXAMPLE, IL IS ROUNDED UP TO 128 (1 BLOCK) IF IL IS BETWEEN 1 AND 127. IL IS ROUNDED UP TO 256 (2 BLOCKS) IF IL = 129 TO 255.	NO ACTION.	NO ACTION.
2	IL IS IGNORED AND THE TRUE FILE DEFINED RECORD LENGTH IS USED.	IL IS IGNORED AND THE TRUE FILE DEFINED RECORD LENGTH IS USED.	NO ACTION.
>2	THE ROUTINE WILL WRITE EXACTLY IL WORDS.	A ZERO LENGTH RECORD IS WRITTEN.	AN END-OF-FILE IS WRITTEN.

WRITF-NUM PARAMETER

NUM IS THE RANDOM ACCESS RECORD NUMBER. USED WITH TYPE 1 AND TYPE 2 FILES.

EXAMPLES

1. NUM = 0, TRANSFER STARTS AT CURRENT RECORD (5)
2. NUM = +n, TRANSFER STARTS AT RECORD NUMBER n (IF NUM = 6, TRANSFER STARTS AT 6)
3. NUM = -n, POSITION n RECORDS BACK (IF NUM = -3, TRANSFER STARTS AT 2)



RWNDF

THIS ROUTINE REWINDS TYPE O FILES, AND SETS DISC FILES TO THE FIRST RECORD IN THE FILE

FORTRAN CALL

```
DIMENSION IDCB (144 + n)
    •
    •
    •
CALL RWNDF (IDCB, IERR)
```

OPTIONAL: IERR

READF

THIS ROUTINE READS A RECORD FROM THE FILE CURRENTLY OPEN TO THE DCB, TO THE USER BUFFER.

FORTRAN CALL

```
DIMENSION IBUF (144+n), IBUF (IL)
      ⋮
CALL READF (IDCB, IERR, IBUF, IL, LEN, NUM)
```

OPTIONAL: IL, LEN, NUM

IL = REQUEST LENGTH
LEN = NUMBER OF WORDS ACTUALLY TRANSFERRED TO IBUF
NUM = RANDOM ACCESS POSITIONING

READ REQUEST FILE LENGTH (IL) vs. FILE TYPES

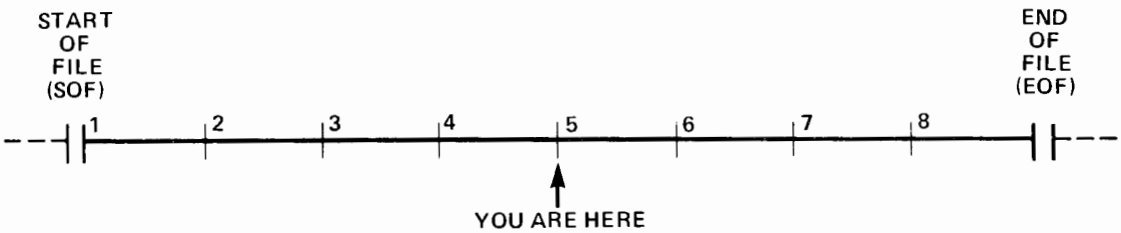
FILE TYPE \ REQUEST LENGTH	IL > 0
0	THE ROUTINE WILL TRANSFER UP TO IL WORDS. IF THE RECORD LENGTH IS SMALLER THAN IL, ONLY THE RECORD IS TRANSFERRED.
1	THE ROUTINE READS EXACTLY IL WORDS. THEREFORE, SINCE THE RECORD LENGTH IS FIXED AT 128 WORDS, LESS THAN A FULL RECORD OR MORE THAN ONE RECORD MAY BE TRANSFERRED. ALL OF IL, HOWEVER, MUST BE WITHIN THE FILE.
>1	THE ROUTINE WILL TRANSFER UP TO IL WORDS. IF THE RECORD LENGTH IS SMALLER THAN IL, ONLY THE RECORD IS TRANSFERRED.

READF-NUM PARAMETER

NUM IS THE RANDOM ACCESS RECORD NUMBER. USED WITH TYPE 1 AND TYPE 2 FILES.

EXAMPLES

1. NUM = O, TRANSFER STARTS AT CURRENT RECORD (5)
2. NUM = +n, TRANSFER STARTS AT RECORD NUMBER n (IF NUM = 6, TRANSFER STARTS AT 6)
3. NUM = -n, POSITION n RECORDS BACK (IF NUM = -3, TRANSFER STARTS AT 2)



POSNT

THIS ROUTINE CAUSES THE NEXT READ OR WRITE TO ACCESS THE GIVEN RECORD IN ANY FILE TYPE.

FORTRAN CALL

```
DIMENSION IDCB (144+n)
    ⋮
CALL POSNT (IDCB, IERR, NUR, IR)
```

OPTIONAL: IR

NUR (*a*)

THE NUR PARAMETER (NEW RECORD) IS DEFINED AS FOLLOWS:

NUR > 0	FORWARD POSITIONING
NUR < 0	BACKWARD POSITIONING
NUR = 0	NO OPERATION

IR (*b*)

THE IR PARAMETER (ABSOLUTE VS RELATIVE IS DEFINED AS FOLLOWS:

IR = 0	THE NUMBER INDICATED BY NUR IS THE NUMBER OF RECORDS TO BE SKIPPED FROM THE PRESENT POSITION TO REACH THE DESIRED RECORD.
IR = 1	THE NUMBER INDICATED BY NUR IS THE ABSOLUTE RECORD NUMBER DESIRED.

- TYPE O FILES MAY BE POSITIONED USING THE POSNT CALL. MOTION MUST BE LEGAL FOR THE DEVICE.
- TYPE 3 AND ABOVE FILES ARE POSITIONED LIKE MAG TAPE FILES.

CLOSE

THIS ROUTINE CLOSES THE DCB AND MAKES THE FILE AVAILABLE TO OTHER CALLERS.

FORTRAN CALL

```
DIMENSION IDCB (144+n)
    •
    •
    •
CALL CLOSE (IDCB, IERR, ITRUN)
```

OPTIONAL: IERR, ITRUN

ITRUN (*n*)

+n = NUMBER OF BLOCKS TO BE DELETED FROM THE END OF THE FILE WHEN IT IS CLOSED.

-n = RETAIN MAIN FILE, DELETE EXTENTS

n = o = STANDARD CLOSE

- IF THE NUMBER OF BLOCKS TO BE DELETED EQUALS THE NUMBER OF BLOCKS IN THE FILE THEN THE FILE IS PURGED.

OPEN

THIS ROUTINE CLOSES THE DCB (IF OPEN) AND THEN OPENS THE NAMED FILE.

FORTRAN CALL

```
DIMENSION IDCB (144+n), NAME (3)
    •
    •
    •
CALL OPEN (IDCB, IERR, NAME, IOPTN, ISECU, ICR, IDCBS)
```

OPTIONAL: IOPTN, ISECU, ICR, IDCBS

OPEN

IOPTN PARAMETER-DISC FILE

T U E

Ø = USE DEFINED FILE TYPE

1 = FORCE TO TYPE 1 FILE

Ø = NON-UPDATE OPEN

1 = UPDATE OPEN

Ø = EXCLUSIVE OPEN

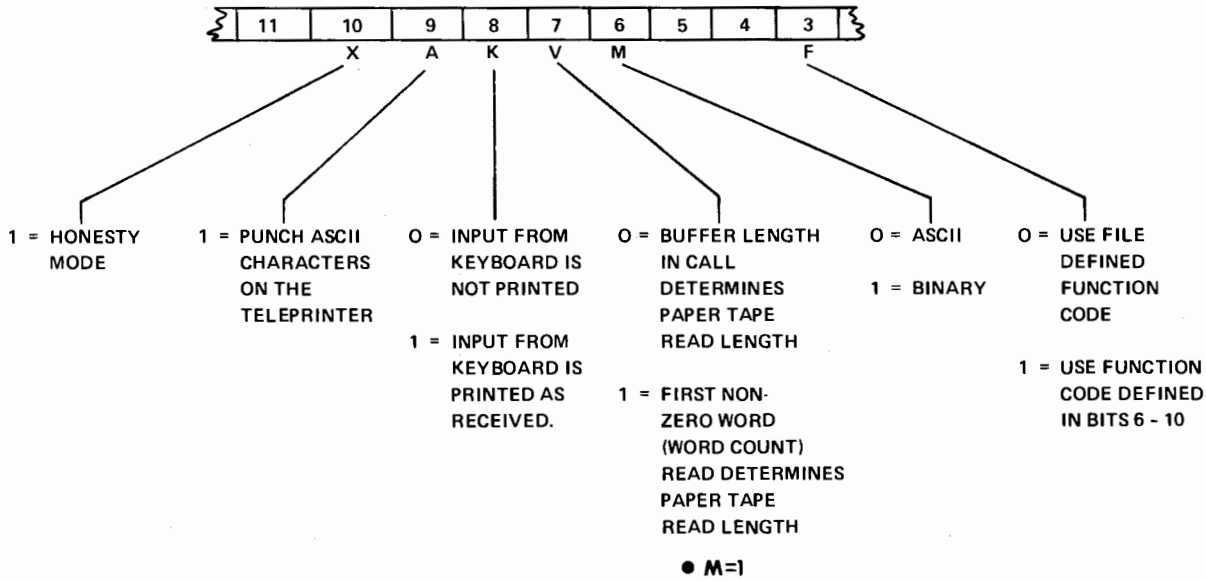
1 = NON-EXCLUSIVE OPEN

- UPDATE OPEN IMPLIES THAT A BLOCK IS READ BEFORE IT IS MODIFIED SO THAT EXISTING RECORDS WITHIN THE BLOCK WILL NOT BE DESTROYED WHEN THE BLOCK IS RE-WRITTEN TO THE DISC. TYPE 1 FILES ARE, BY DEFINITION, UPDATE FILES. TYPE 2 AND ABOVE FILES MUST BE OPENED AS UPDATE FILE BEFORE MODIFYING EXISTING RECORDS.

- A FILE MAY BE OPENED EXCLUSIVELY TO ONE PROGRAM OR NON-EXCLUSIVELY TO AS MANY AS SEVEN PROGRAMS.

OPEN

IOPTN PARAMETER-TYPE Ø FILE



LOCF

THIS ROUTINE RETURNS THE LOCATION AND STATUS OF THE NEXT SEQUENTIAL RECORD. THE INFORMATION IS OBTAINED FROM THE DCB.

FORTRAN CALL

```
DIMENSION IDCB (144 + n)
        •
        •
        •
CALL LOCF (IDCB, IERR, IREC, IRB, IOFF, JSEC, JLU, JTY, JREC)
```

OPTIONAL: IRB, IOFF, JSEC, JLU, JTY, JREC

IREC THE NUMBER OF THE NEXT RECORD.

IRB THE RELATIVE BLOCK OF THE NEXT RECORD.

IOFF THE BLOCK OFFSET OF THE NEXT RECORD. $0 < IOFF < 128$

JSEC THE NUMBER OF SECTORS IN THE MAIN FILE.

JLU THE LOGICAL UNIT THE FILE IS ON (DISC OR NON-DISC).

JTY THE FILE TYPE SET IN THE DCB.

JREC THE RECORD SIZE WORD FOR DISC FILES. READ/WRITE CODE FOR TYPE 0 FILES. BIT 15 = 1 = READ LEGAL, BIT 0 = 1 = WRITE LEGAL.

APOSN

THIS ROUTINE IS USED TO RANDOMLY ACCESS SEQUENTIAL FILES.

FORTRAN CALL

```
DIMENSION IDCB (144 + n)
      •
      •
      •
CALL APOSN (IDCB, IERR, IREC, IRB, IOFF)
```

OPTIONAL: IRB, IOFF

- TYPICALLY, A DIRECTORY CONTAINING THE LOCATION INFORMATION OF EACH RECORD IN THE FILE WOULD BE BUILT BY THE USER. THIS WOULD ALLOW THE USE OF APOSN FOR RANDOM ACCESS. IREC, IRB AND IOFF MAY BE OBTAINED USING THE LOCF CALL.

NAMF

THIS ROUTINE CLOSES THE DCB AND THEN RENAMES THE SPECIFIED FILE.

FORTRAN CALL

```
DIMENSION IDCB (144+n), NAME (3), NNAME (3)
:
CALL NAMF (IDCB, IERR, NAME, NNAME, ISECU, ICR)
```

OPTIONAL: ISECU, ICR

PURGE

THIS ROUTINE CLOSES THE DCB (IF OPEN), AND DELETES THE NAMED FILE AND ALL ITS EXTENTS.

FORTRAN CALL

```
DIMENSION IDCB (144+n), NAME (3)
      •
      •
      •
CALL PURGE (IDCB, IERR, NAME, ISECU, ICR)
```

OPTIONAL: ISECU, ICR

- THIS ROUTINE WILL NOT PURGE TYPE Ø FILES.

FCONT

THIS ROUTINE SENDS THE STANDARD RTE I/O CONTROL REQUESTS TO TYPE O (NON DISC) FILES.

FORTRAN CALL

```
DIMENSION IDCB (144+n)
      ⋮
CALL FCONT (IDCB, IERR, ICON1, ICON2)
```

OPTIONAL: ICON2

ICON 1 *(conwd)*

- THE CONTROL WORD VALUE *(conwd)* HAS ONE FIELD THAT IS ORED WITH THE DEVICE LOGICAL UNIT NUMBER

15	14	13	12	11	10 9	8 7 6	5 4 3	2 1 0
0	0	0	0	0	FUNCTION CODE		NOT USED	

FUNCTION CODE (OCTAL)	ACTION
00	UNUSED
01	WRITE END-OF-FILE (MAG TAPE)
02	BACKSPACE ONE RECORD (MAG TAPE)
03	FORWARD SPACE ONE RECORD (MAG TAPE)
04	REWIND (MAG TAPE)
05	REWIND STANDBY (MAG TAPE)
06	DYNAMIC STATUS (MAG TAPE)
07	SET END-OF-PAPER TAPE
10	GENERATE PAPER TAPE LEADER
11	LIST OUTPUT LINE SPACING
12	WRITE GAP (MAG TAPE)
13	FORWARD SAPCE FILE (MAG TAPE)
14	BACKWARD SPACE FILE (MAG TAPE)

- THE FOLLOWING FUNCTIONS ARE DEFINED FOR DVR00, DVR01 AND DVR02:

20 ENABLE TERMINAL – ALLOWS TERMINAL TO SCHEDULE ITS PROGRAM WHEN ANY KEY IS STRUCK.

21 DISABLE TERMINAL – INHIBITS SCHEDULING OF TERMINAL'S PROGRAM.

22 SET TIME OUT – THE OPTIONAL THIRD PARAMETER IS SET AS THE NEW TIME OUT INTERVAL.

23 IGNORE ALL FURTHER ACTION REQUESTS UNTIL:
a) THE QUEUE IS EMPTY
b) AN INPUT REQUEST IS RECEIVED
c) A RESTORE CONTROL REQUEST IS RECEIVED

24 RESTORE OUTPUT PROCESSING (THIS REQUEST IS USUALLY NOT NEEDED).

ICON2 *(n)*

- FUNCTION CODE 11_8 (LIST OUTPUT LINE SPACING), REQUIRES THE OPTIONAL PARAMETER ICON2. ICON2 MUST DESIGNATE THE NUMBER OF LINES TO BE SPACED ON THE SPECIFIED LOGICAL UNIT. A NEGATIVE PARAMETER SPECIFIES A PAGE EJECT ON A LINE PRINTER. FOR DETAILS OF LINE PRINTER FORMATTING CONSULT APPENDIX F IN THE RTE MANUAL.

FSTAT

PURPOSE

THIS ROUTINE RETURNS INFORMATION ON ALL CARTRIDGE LABELS IN THE SYSTEM.

FORTRAN CALL

```
DIMENSION ISTAT (125)
    ●
    ●
    ●
CALL FSTAT (ISTAT)
```

ISTAT

THE CARTRIDGE STATUS IS CONTAINED IN FOUR WORD ENTRIES

WORD

0	LOGICAL UNIT NUMBER (FIRST DISC)
1	LAST TRACK FOR FMP
2	CARTRIDGE LABEL
3	LOCKING PROGRAM'S ID SEGMENT ADDRESS, OR O IF NOT LOCKED.
4	LOGICAL UNIT NUMBER (SECOND DISC)
●	
●	
●	
124	O

- **THIS LIST IS TERMINATED WITH A ZERO.**

POST

THIS ROUTINE POSTS THE DCB BUFFER ON THE DISC, IF NEEDED, AND SETS FLAGS INDICATING NO DATA IN CORE. IF ONE PROGRAM IN A GROUP OF CO-OPERATING PROGRAMS UPDATES A RECORD AND THEN DOES NOT POST THE DCB, ANOTHER PROGRAM SUBSEQUENTLY ACCESSING THAT RECORD WILL NOT "SEE" THE UPDATE, IE, HIS DCB IS FILLED WITH THE OLD RECORD FROM THE DISC.

FORTRAN CALL:

```
DIMENSION IDCB (144 + n)
    ⋮
CALL POST (IDCB, IERR)
```

THE POST ROUTINE IS USED WITH THE SYSTEM RNRQ REQUEST TO ALLOW SEVERAL PROGRAMS TO MODIFY A FILE WITHOUT REQUIRING EXCLUSIVE OPENS. THE SUGGESTED SEQUENCE IS AS FOLLOWS:

1. CALL OPEN
2. READ FILE TO PICK UP RN NUMBER (IT IS ASSUMED TO BE IN THE FILE).
3. CALL POST TO CLEAR IN-CORE FLAGS.
4. CALL RNRQ TO LOCK THE FILE.
5. READ THE RECORD TO BE MODIFIED.
6. MODIFY THE RECORD AND WRITE IT OUT (THE DATA MAY OR MAY NOT BE POSTED).
7. CALL POST TO POST ANY DATA NOT ALREADY POSTED (WRITTEN TO THE FILE).
8. CALL RNRQ TO UNLOCK THE RN.

IDCBS

TO DETERMINE THE SIZE OF THE DCB ACTUALLY USED.

FORTRAN CALL

```
DIMENSION IDCB (144 + n)

ISIZE = IDCBS (IDCB)
```

- ISIZE WILL CONTAIN NUMBER OF WORDS ACTUALLY USED.

USED WHEN A PROGRAM MUST DETERMINE ACTUAL SIZE OF DCB SET UP BY SYSTEM. ALTHOUGH THE CREATE AND OPEN CALLS ALLOW THE USER TO DECLARE AN DCB LARGER THAN 144 WORDS, THE SYSTEM MAY NOT USE THE ENTIRE DCB, DEPENDING UPON FILE SIZE. ANY LEFT OVER AREA MAY BE USED FOR ANOTHER FILE'S DCB.

EXERCISE

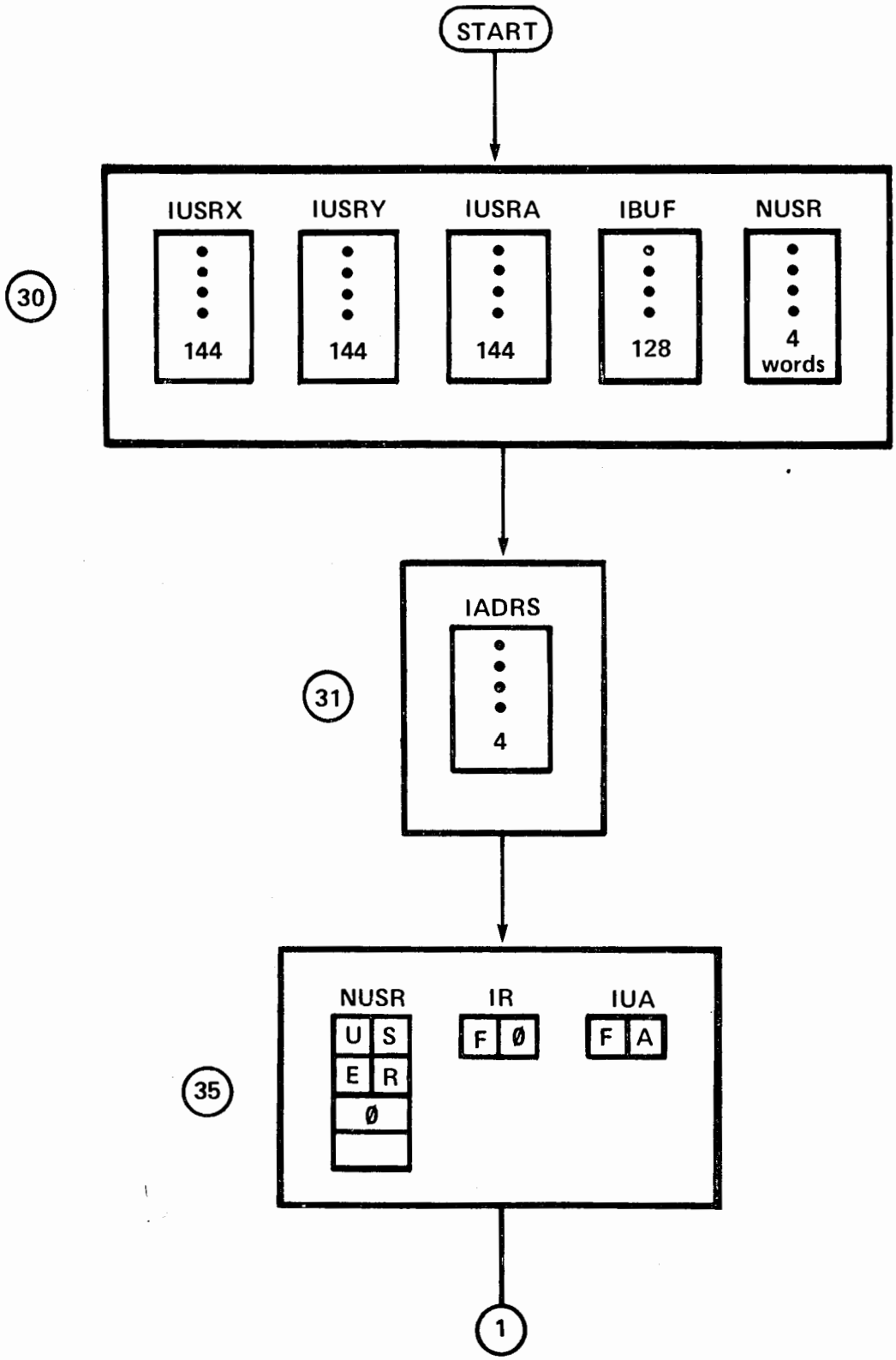
WRITE A PROGRAM TO:

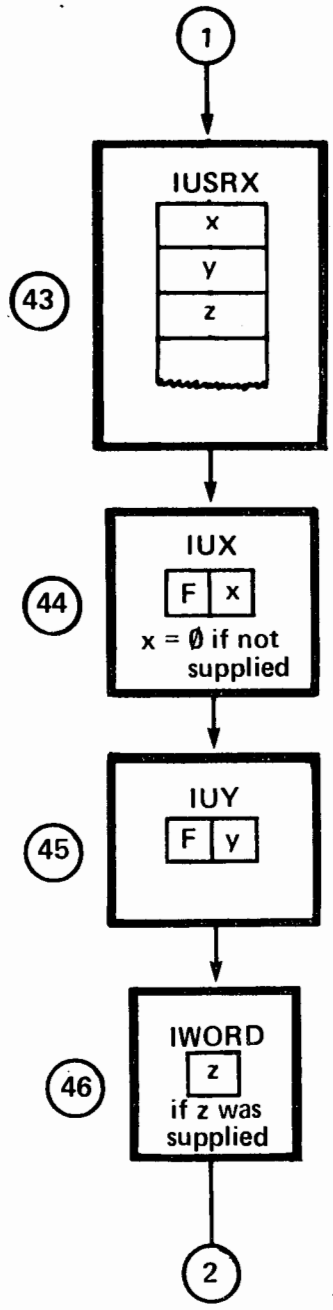
1. REQUEST A FILE NAME FROM OPERATOR.
2. CREATE A TYPE 2 FILE TO ACCOMMODATE 128 1-WORD RECORDS.
3. REQUEST A RECORD NUMBER FROM THE OPERATOR (RANGE 1-128).
4. WRITE THE ORDINAL RETURNED BY THE OPERATOR IN THAT RECORD.
5. READ THE RECORD AFTER WRITING IT AND DISPLAY CONTENTS ON CRT.
6. CHECK IERR AFTER EACH FMP CALL – ACTUALLY ALL YOU HAVE TO DO IS DISPLAY THE CONTENTS OF IERR ON THE CRT.
7. ATTEMPT ILLEGAL OPERATIONS SUCH AS WRITE PAST EOF, TOO LARGE A RECORD, ETC. NOTE WHAT APPEARS IN IERR.

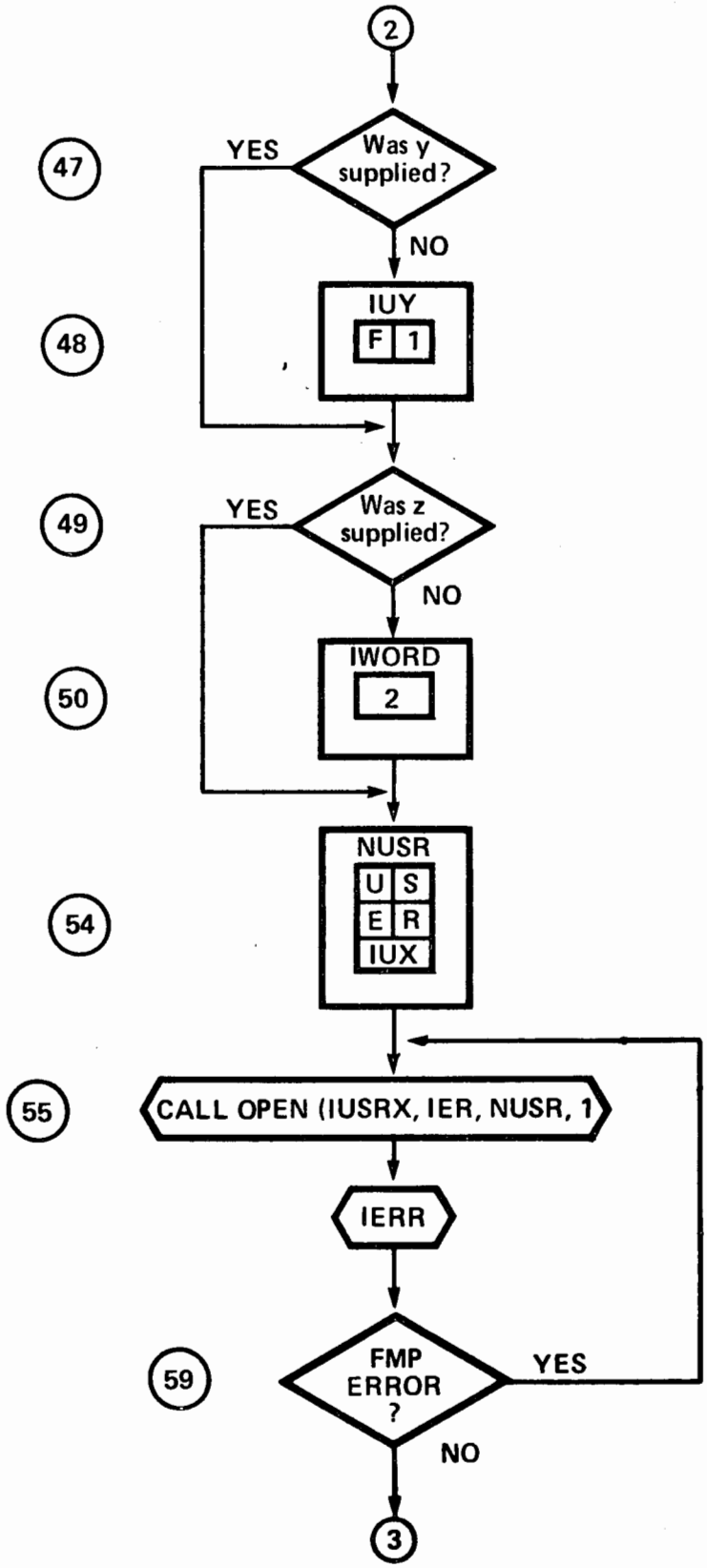
SELF-STUDY EXERCISE

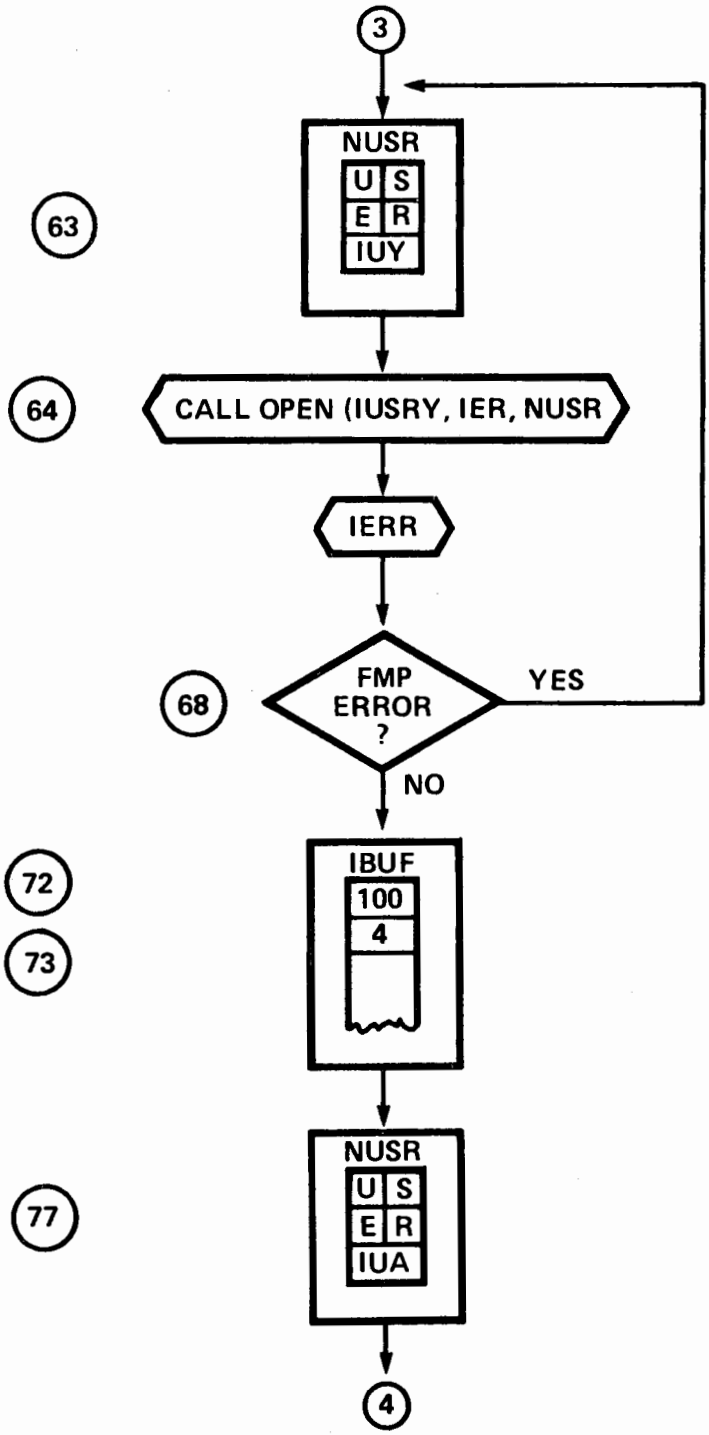
THE FOLLOWING IS A FLOW CHART OF PROGRAM DEMOI, THE CODE FOR WHICH IS IN THE BATCH SPOOL MONITOR PROGRAMMING AND OPERATING MANUAL. USING THESE FLOW CHARTS, STUDY THE PROGRAM FOR UNDERSTANDABILITY OF THE PROGRAM CALLS.

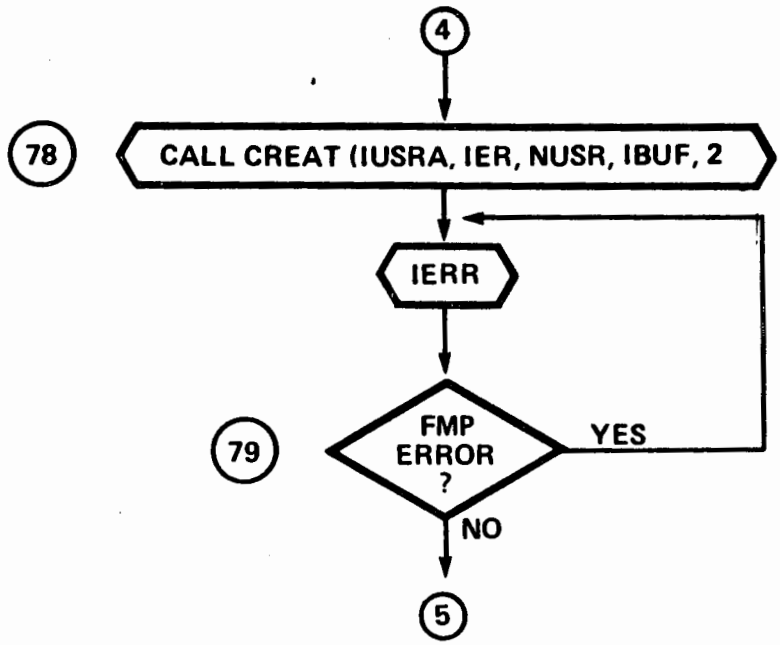
PROGRAM DEMO 1
FLOW CHART

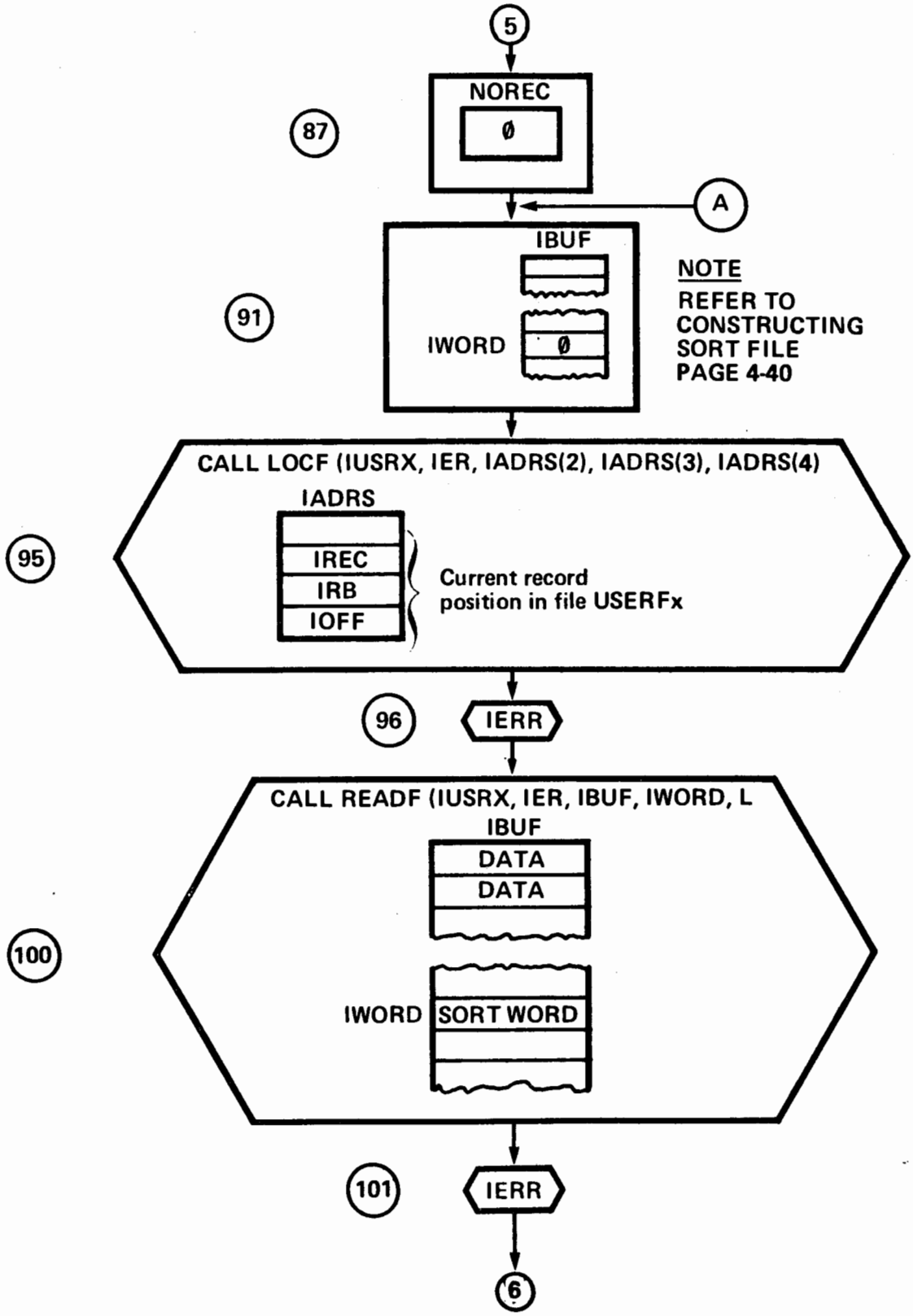

START
30
IUSRX
144
IUSRY
144
IUSRA
144
IBUF
128
NUSR
4
words
31
IADRS
4
35
NUSR
U
S
E
R
Ø
IR
F
Ø
IUA
F
A
1

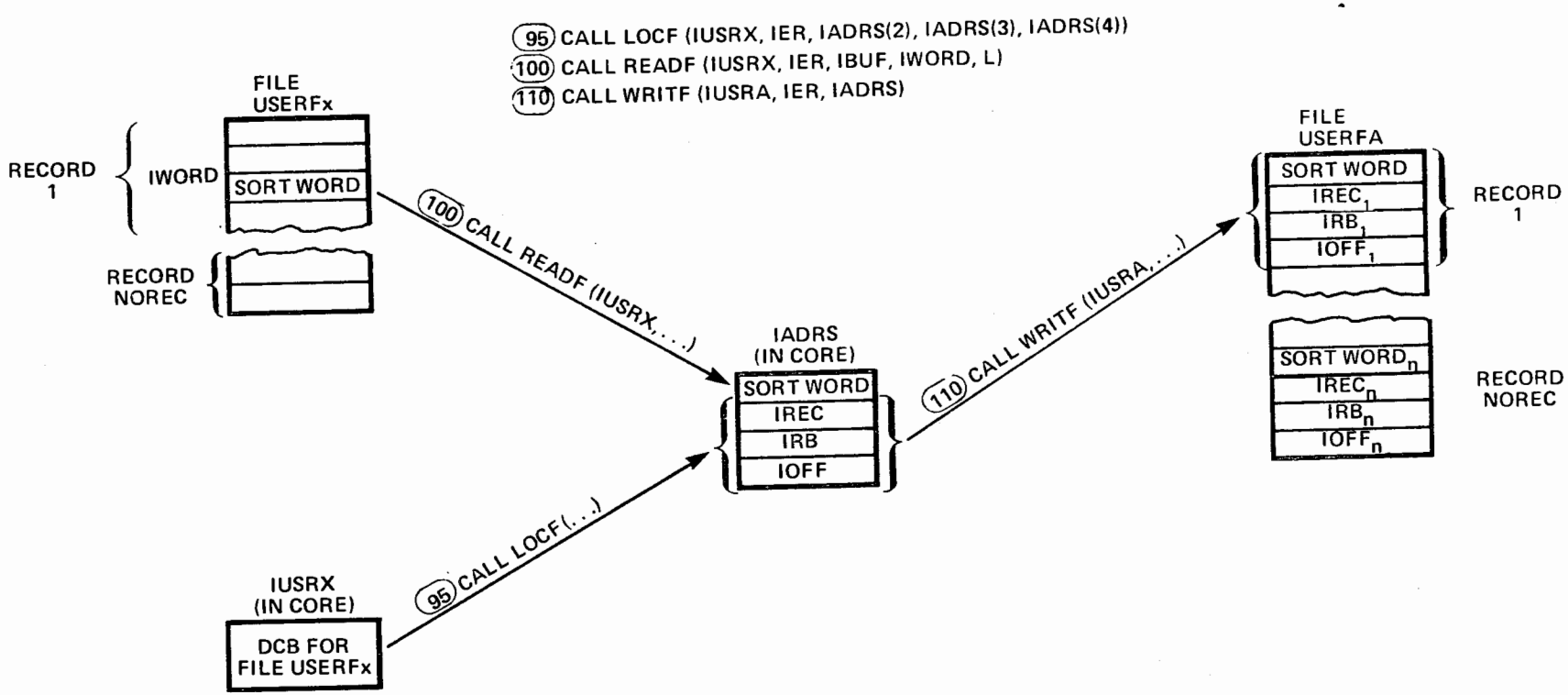

1
IUSRX
x
y
z
43
IUX
F
x
x = Ø if not supplied
44
IUY
F
y
45
IWORD
z
if z was supplied
46
2


2
47
YES
Was y supplied?
NO
48
IUY
F 1
49
YES
Was z supplied?
NO
50
IWORD
2
54
NUSR
U S
E R
IUX
55
CALL OPEN (IUSRX, IER, NUSR, 1)
IERR
59
FMP ERROR ?
YES
NO
3

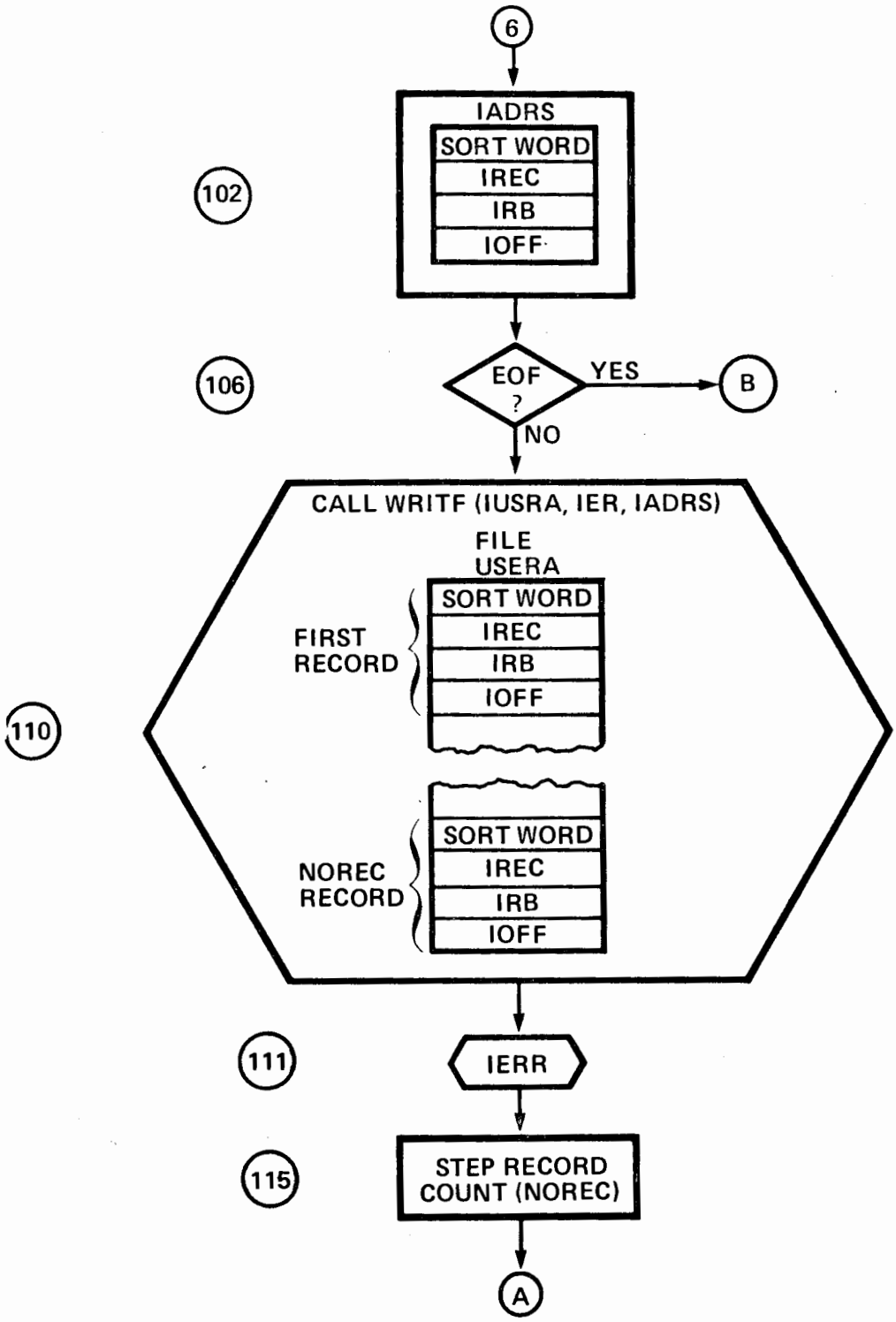

3
NUSR
U S
E R
IUY
63
64
CALL OPEN (IUSRY, IER, NUSR
IERR
68
FMP
ERROR
?
YES
NO
72
73
IBUF
100
4
NUSR
U S
E R
IUA
77
4

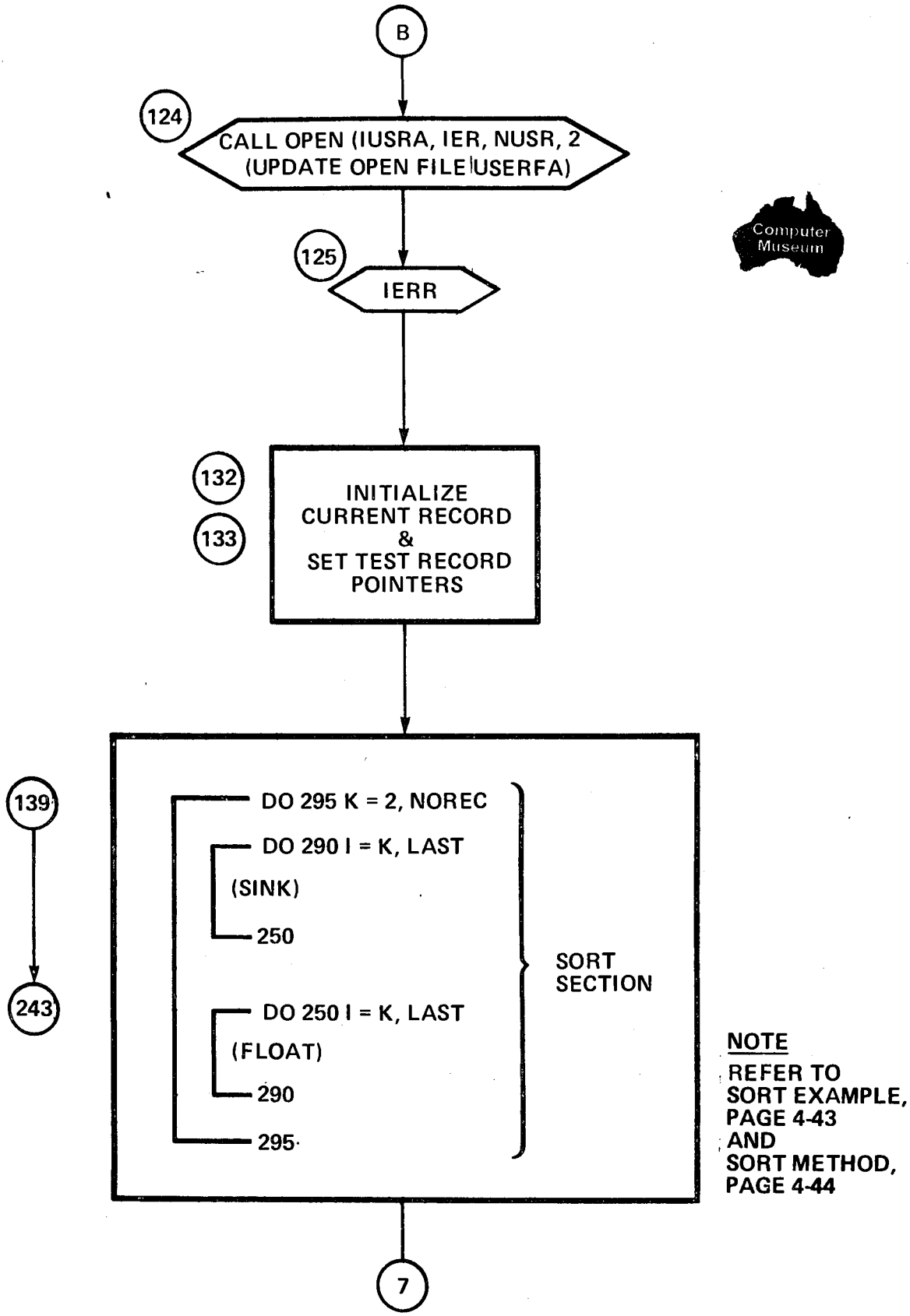

4
78
CALL CREAT (IUSRA, IER, NUSR, IBUF, 2
IERR
79
FMP ERROR ?
YES
NO
5


5
NOREC
Ø
87
A
IBUF
91
IWORD
Ø
NOTE
REFER TO
CONSTRUCTING
SORT FILE
PAGE 4-40
CALL LOCF (IUSRX, IER, IADRS(2), IADRS(3), IADRS(4)
IADRS
95
IREC
IRB
IOFF
Current record
position in file USERFx
96
IERR
CALL READF (IUSRX, IER, IBUF, IWORD, L
IBUF
DATA
DATA
100
IWORD
SORT WORD
101
IERR
6



CONSTRUCTING SORT FILE


6
IADRS
SORT WORD
IREC
IRB
IOFF
102
106
EOF ?
YES
B
NO
CALL WRITF (IUSRA, IER, IADRS)
FILE
USERA
FIRST RECORD
SORT WORD
IREC
IRB
IOFF
110
NOREC RECORD
SORT WORD
IREC
IRB
IOFF
111
IERR
115
STEP RECORD COUNT (NOREC)
A



DO 295 K = 2, NOREC (SORT ROUTINE)

DO 250 I = K, LAST (SINK ROUTINE)

LAST = 8

5	4						4
4	5	3					3
3		5	5				5
6			6	6			6
7				7	2		2
2					7	1	1
1						7	7	7
8							8	8

I =	2	3	4	5	6	7	8
ISWCO =	1	1	1	1	1	1	1

LAST = 7

1							1
4	3						3
3	4	4					4
5		5	5				5
6			6	2			2
2				6	6		6
7					7	7	7
8							8

I =	3	4	5	6	7
ISWCO =	1	1	1	1	1

LAST = 6

1						1
2						2
3	3					3
4	4	4				4
5		5	5			5
6				6		6
7						7
8						8

I =	4	5	6
ISWCO =	0	0	0

DO 290 I = K, LAST (FLOAT ROUTINE)

4						1	1
3					1	4	4
5				1	3		3
6			1	5			5
2		1	6				6
1	1	2					2
7	7	7					7
8	8						8

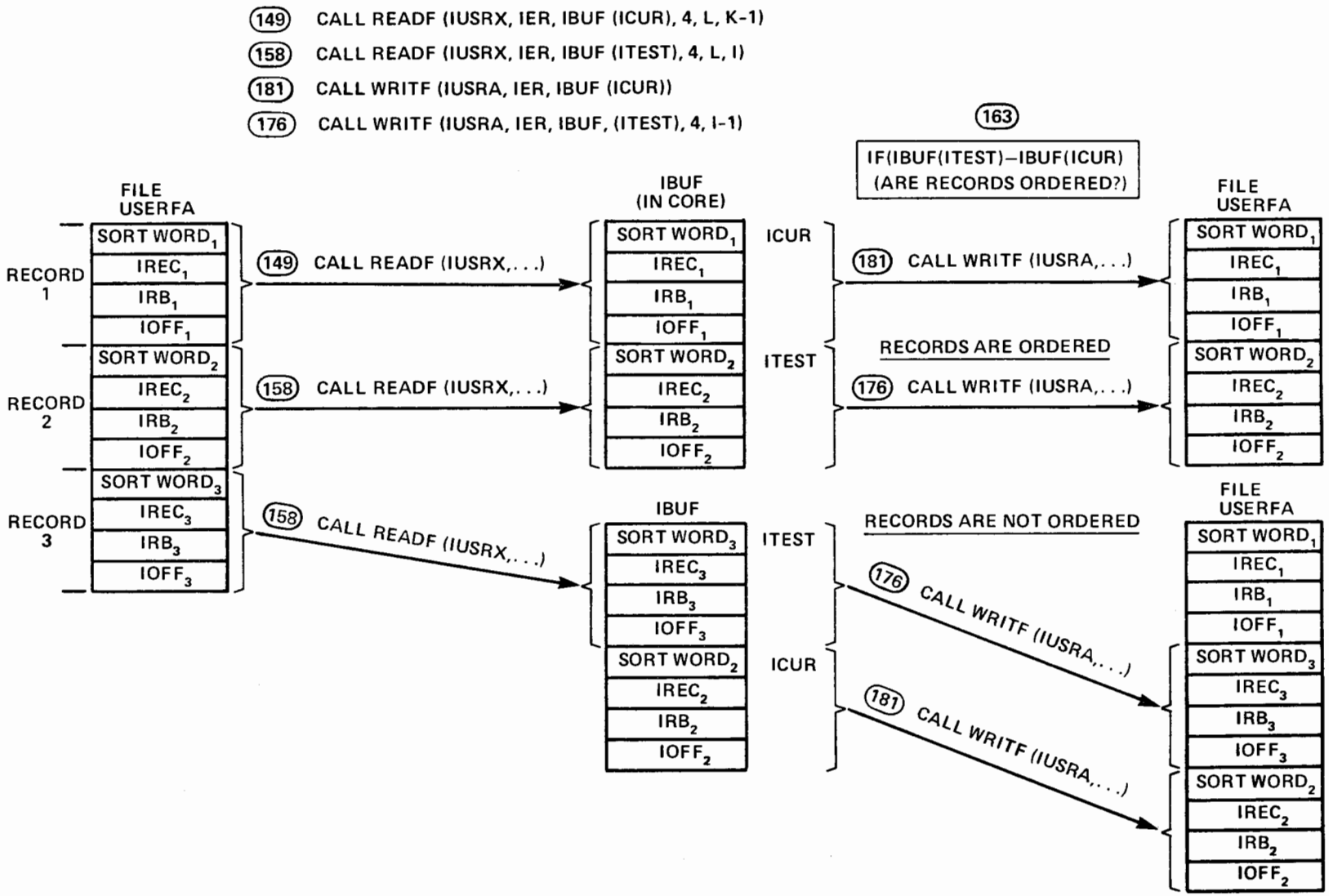
I =	2	3	4	5	6	7	8
ISWCO =	0	0	1	1	1	1	1
IREC =	7	6	5	4	3	2	1

1							1
3						2	2
4					2	3	3
5				2	4		4
2			2	5			5
6	6	6					6
7	7						7
8							8

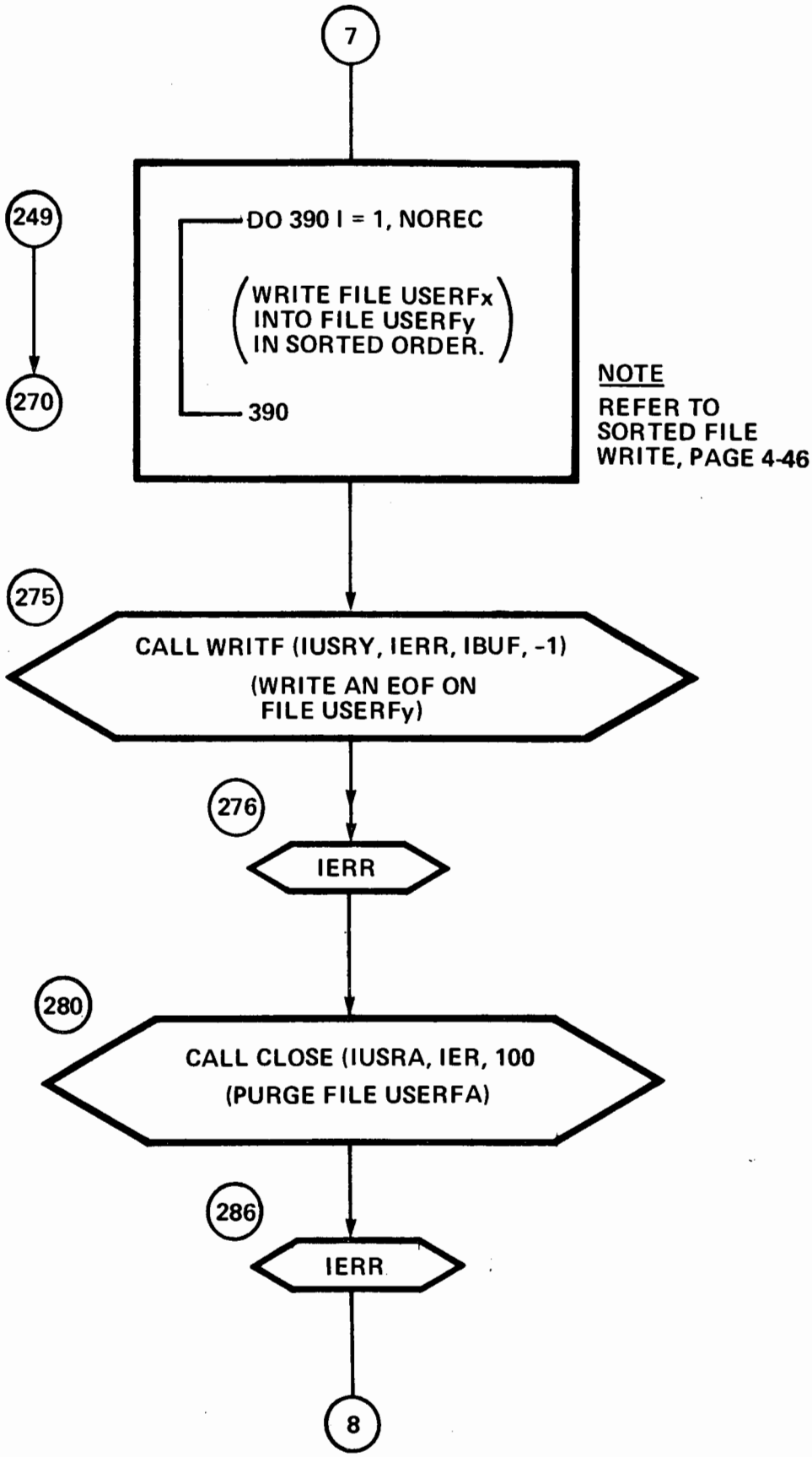
I =	3	4	5	6	7
ISWCO =	0	0	1	1	1
IREC =	6	5	4	3	2

SORT EXAMPLE

CONTENTS OF FILE USERFA WHILE BEING SORTED


149 CALL READF (IUSRX, IER, IBUF (ICUR), 4, L, K-1)
158 CALL READF (IUSRX, IER, IBUF (ITEST), 4, L, I)
181 CALL WRITF (IUSRA, IER, IBUF (ICUR))
176 CALL WRITF (IUSRA, IER, IBUF, (ITEST), 4, I-1)
163
IF(IBUF(ITEST)–IBUF(ICUR)
(ARE RECORDS ORDERED?)
FILE USERFA
RECORD 1
RECORD 2
RECORD 3
SORT WORD1
IREC1
IRB1
IOFF1
SORT WORD2
IREC2
IRB2
IOFF2
SORT WORD3
IREC3
IRB3
IOFF3
149 CALL READF (IUSRX,. . .)
158 CALL READF (IUSRX,. . .)
158 CALL READF (IUSRX,. . .)
IBUF (IN CORE)
ICUR
ITEST
IBUF
ITEST
ICUR
181 CALL WRITF (IUSRA,. . .)
RECORDS ARE ORDERED
176 CALL WRITF (IUSRA,. . .)
RECORDS ARE NOT ORDERED
176 CALL WRITF (IUSRA,. . .)
181 CALL WRITF (IUSRA,. . .)
FILE USERFA
FILE USERFA

SORT METHOD

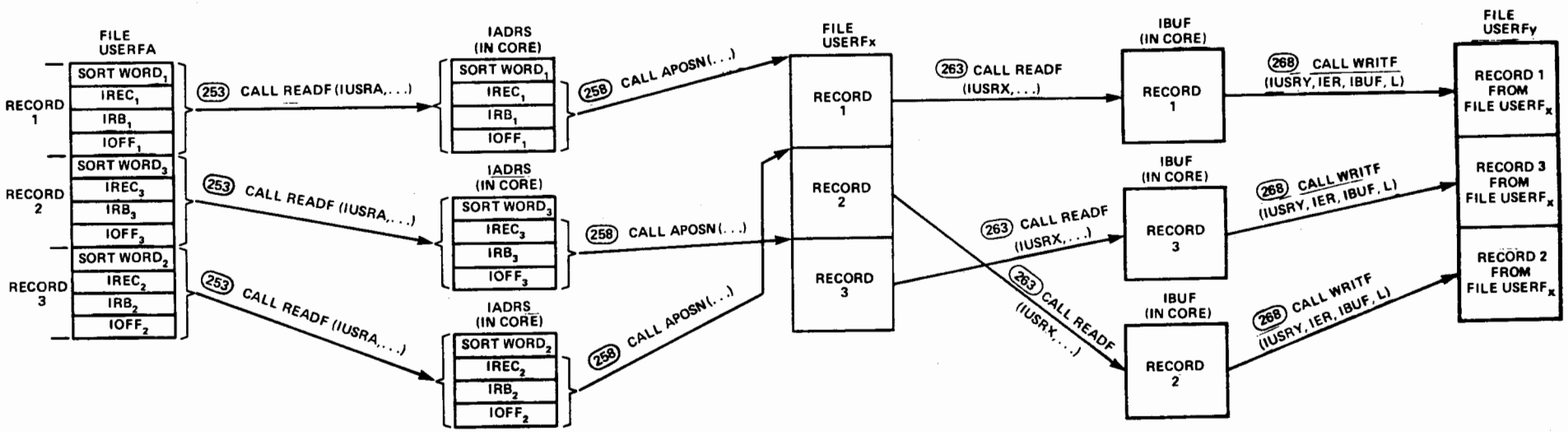

7
249
DO 390 I = 1, NOREC
WRITE FILE USERFx
INTO FILE USERFy
IN SORTED ORDER.
270
390
NOTE
REFER TO
SORTED FILE
WRITE, PAGE 4-46
275
CALL WRITF (IUSRY, IERR, IBUF, -1)
(WRITE AN EOF ON
FILE USERFy)
276
IERR
280
CALL CLOSE (IUSRA, IER, 100
(PURGE FILE USERFA)
286
IERR
8

253 CALL READF (IUSRX, IER, IADRS, 4, L, I)

258 CALL APOSN (IUSRX, IER, IADRS(2), IADRS(3), IADRS(4))

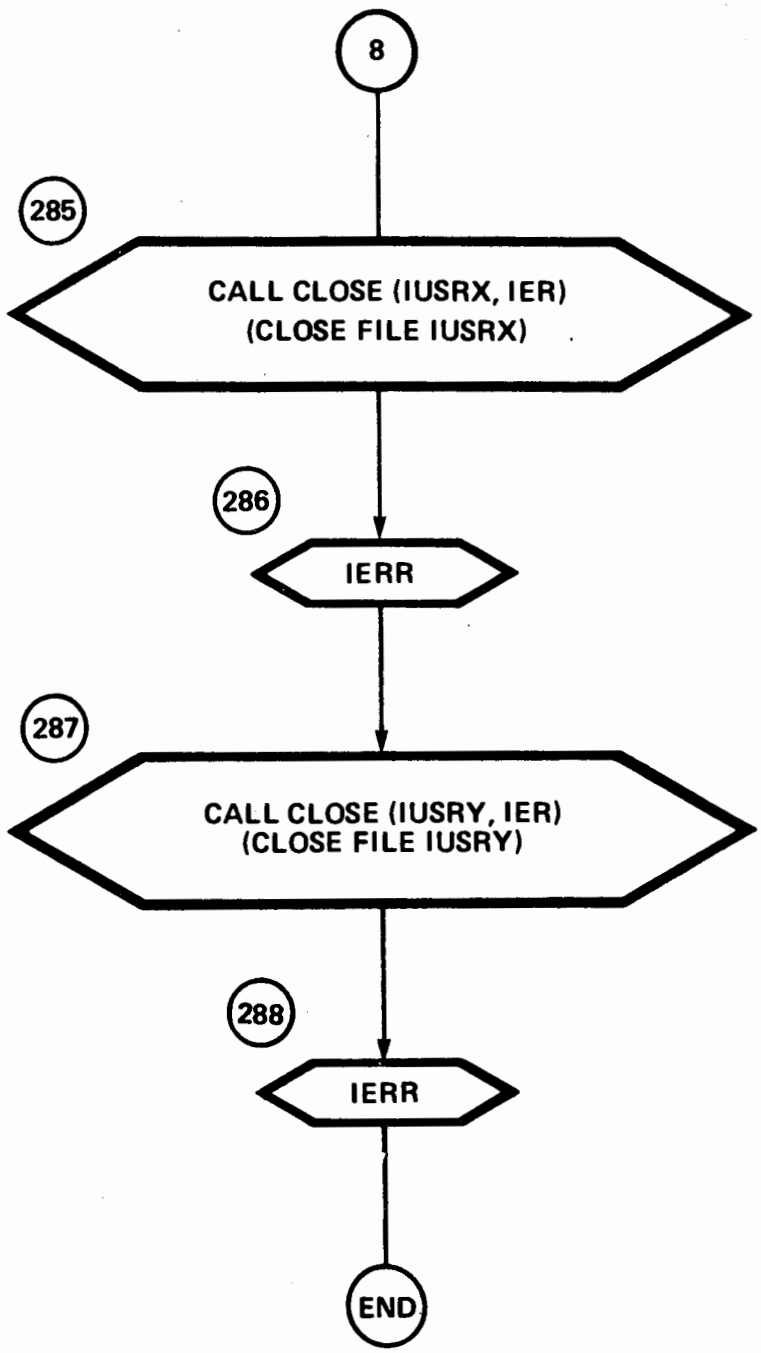
263 CALL READF (IUSRX, IER, IBUF, 128, L)

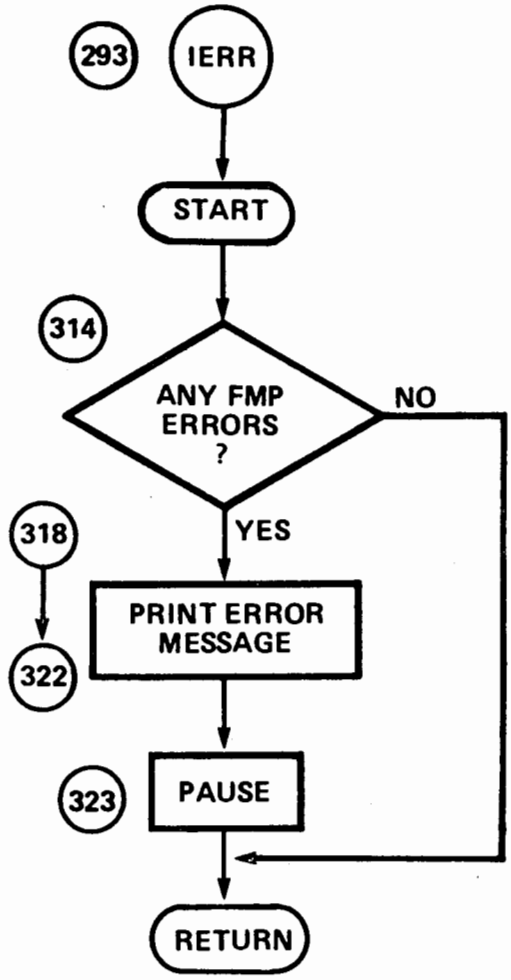
268 CALL WRITF (IUSRY, IER, IBUF, L)



SORTED FILE WRITE

WRITE FILE USERFx INTO FILE USERFy IN SORTED ORDER


8
285
CALL CLOSE (IUSRX, IER)
(CLOSE FILE IUSRX)
286
IERR
287
CALL CLOSE (IUSRY, IER)
(CLOSE FILE IUSRY)
288
IERR
END


293
IERR
START
314
ANY FMP ERRORS ?
NO
YES
318
PRINT ERROR MESSAGE
322
323
PAUSE
RETURN

RTE-II SYSTEM GENERATION CONSIDERATIONS FOR BATCH/SPOOL

FMP INSTALLATION

FMP SYSTEM INSTALLATION

THE FMP IS INSTALLED IN THE RTE SYSTEM IN TWO PHASES.

1. INCORPORATION OF THE FMP MODULES.
   INTO THE SYSTEM AT GENERATION TIME.

2. INITIALIZATION OF THE SYSTEM AND AUXILIARY DISCS.

● *NO PROVISIONS EXIST FOR ON-LINE LOADING OF THE FMP MODULES.*

INSTALLATION CONSIDERATION

- THE FMP MUST BE CONFIGURED INTO THE RTE SYSTEM DURING THE PROGRAM INPUT PHASE OF RTGEN.
  - THE FMP MODULES ARE NUMBERED ACCORDING TO THE ORDER OF THEIR LOADING.
- FMGR HAS A PRIORITY OF 90, IS SEGMENTED INTO NINE PARTS, AND REQUIRES 5K OF BACKGROUND AREA.
- D.RTR HAS A PRIORITY OF 1, REQUIRES 1K OF AREA AND IS SUPPLIED AS A FOREGROUND DISC RESIDENT PROGRAM.
  - IF SPACE PERMITS, IT IS RECOMMENDED THAT IT BE MADE CORE RESIDENT FOR GREATER SPEED.
  - D.RTR MUST HAVE A HIGHER PRIORITY THAN ANY PROGRAM USING THE FMP.

- *PROGRAM TYPE AND PRIORITY OF D.RTR MAY BE CHANGED DURING THE PARAMETER INPUT PHASE OF RTGEN.*
- *ALL STARTING TRACK NUMBERS MUST BE THE SAME IF THE CARTRIDGES ARE TO BE EXCHANGED LU TO LU OR SYSTEM TO SYSTEM. THIS IS BECAUSE FMP USES RELATIVE TRACK NUMBERS IN THE DIRECTORY.*

FMGR INITIALIZATION

EACH TIME THE RTE SYSTEM IS LOADED FROM THE DISC FMGR IS SCHEDULED. IF FMP HAS NOT BEEN INITIALIZED TO THE SYSTEM FMGR PRINTS ON THE SYSTEM TTY:

```
FMGR ØØ2
:
```

THE USER MUST THEN INITIALIZE THE SYSTEM DISC (LU2) USING THE IN OPERATOR COMMAND.

AFTER SUCCESSFUL INITIALIZATION OF THE SYSTEM DISC AND THERE IS AN AUXILIARY DISC, FMGR PRINTS ON THE SYSTEM TTY:

```
FMGR ØØ3
:
```

THE USER MUST THEN INITIALIZE THE AUXILIARY DISC. IF NO TRACKS ARE TO BE ASSIGNED TO THE AUXILIARY DISC, THE AUXILIARY DISC, THE RESPONSE IS

```
:IN,SC, -3,Ø
```

AFTER SUCCESSFUL INITIALIZATION, FMGR TERMINATES (NO MESSAGE IS PRINTED) AND CONTROL IS RETURNED TO THE RTE

SPOOL SYSTEM MODULES

NAME	RESIDENCY	SIZE
JOB	FOREGROUND DISC (MAY BE TYPE 1)	3563_8
SMP	FOREGROUND DISC (MAY BE TYPE 1)	5613_8
SPOUT	FOREGROUND CORE (MAY BE TYPE 1)	662_8
GASP	BACKGROUND DISC	$\approx 13000_8$
EXTND	FOREGROUND CORE	226_8
DVS43	DRIVER AREA	2265_8

RTE-II SYSTEM RESOURCES FOR BATCH/SPOOL

* # OF I/O CLASSES?
  (2 REQUIRED)
* # OF LU MAPPINGS?
  (8 RECOMMENDED)
* # OF RESOURCE NUMBERS?
  (4 REQUIRED)

SPOOL EQT'S AND LU'S

EQT 15?	19 = EQT # ?
31, DVS 43, X = 18	15
EQT 16?	20 = EQT # ?
32, DVS 43, X = 18	16
EQT 17?	21 = EQT # ?
33, DVS 43, X = 18	17
EQT 18?	22 = EQT # ?
34, DVS 43, X = 18	18
EQT 19?	23 = EQT # ?
35, DVS 43, X = 18	19
EQT 20?	24 = EQT # ?
36, DVS 43, X = 18	20

- The EQT entry must reference
- The X option for the EQT entry specifies an 18 word extension to the EQT.
- For each spool EQT, there must be a corresponding interrupt table entry, i.e.,

  31, EQT, 15

PROGRAM INITIATION

CALL TRNON (name, itime, ierr)

name = ASCII program name

itime = three word integer array containing absolute time of day in hours, minutes, seconds

ierr = error return parameter

PROGRAM RESCHEDULING

CALL START (name, mult, ires, ierr)

name = ASCII program name

mult = time interval in units

ires = units of time

0: 10 millisecond intervals
1: milliseconds
2: seconds
3: minutes

ierr = error return parameter

TIME DELAY

CALL WAIT (mult, ires, ierr)

mult = time interval in units

ires = units of time

0: 10 millisecond intervals
1: milliseconds
2: seconds
3: minutes

ierr = error return parameter

SEQUENTIAL ANALOG INPUT

CALL AISQ (num, iscan, idata, ierr)

num = number of channels to be read

num < 0: paced scan

num > 0: non-paced scan

iscan = starting channel of scan

iscan > 0: sequential scan

iscan < 0: repeated readings on channel

idata = integer data array

ierr = error return parameter

SEQUENTIAL ANALOG INPUT

CALL AISQF (num, iscan, volts, ierr)

num = number of channels to be read

num < 0: paced scan

num > 0: non-paced scan

iscan = starting channel fo scan

iscan > 0: sequential scan

iscan < 0: repeated readings on channel

volts = real data array

ierr = error return parameter

RANDOM ANALOG INPUT

CALL AIRD (num, ichan, idata, ierr)

num = number of channels to be read

num < 0: paced scan

num > 0: non-paced scan

ichan = integer array of channel numbers

idata = integer data array

ierr = error return parameter

RANDOM ANALOG INPUT

CALL AIRDF (num, ichan, volts, ierr)

num = number of channels to be read

num $<$ 0: paced scan

num $>$ 0: non-paced scan

ichan = integer array of channel numbers

volts = real data array

ierr = error return parameter

ANALOG OUTPUT

CALL AO (num, ichan, idata, ierr)

num = number of output values

num $<$ 0: paced output

num $>$ 0: non-paced output

ichan = integer array of channel numbers

idata = integer data array

ierr = error return parameter

ANALOG OUTPUT

CALL AOF (num, ichan, data, ierr)

num = number of output values

num $<$ 0: paced output

num $>$ 0: non-paced output

ichan = integer array of channel numbers

data = real array of output values

ierr = error return parameter

LLMPX GAIN

CALL SGAIN (ichan, gain)

ichan = analog input channel number

gain = value of gain to be set for ichan
(12.5, 25, 50, 100, 125, 250, 500 or 1000)

CALL RGAIN (ichan, gain)

ichan = analog input channel number

gain = value of gain returned

PACER

CALL PACER (irate, mult, mode, iunit, ierr)

irate = pacer period in microseconds
($0 \leqslant$ irate $\leqslant 255$)

mult = exponent of pace rate multiplier

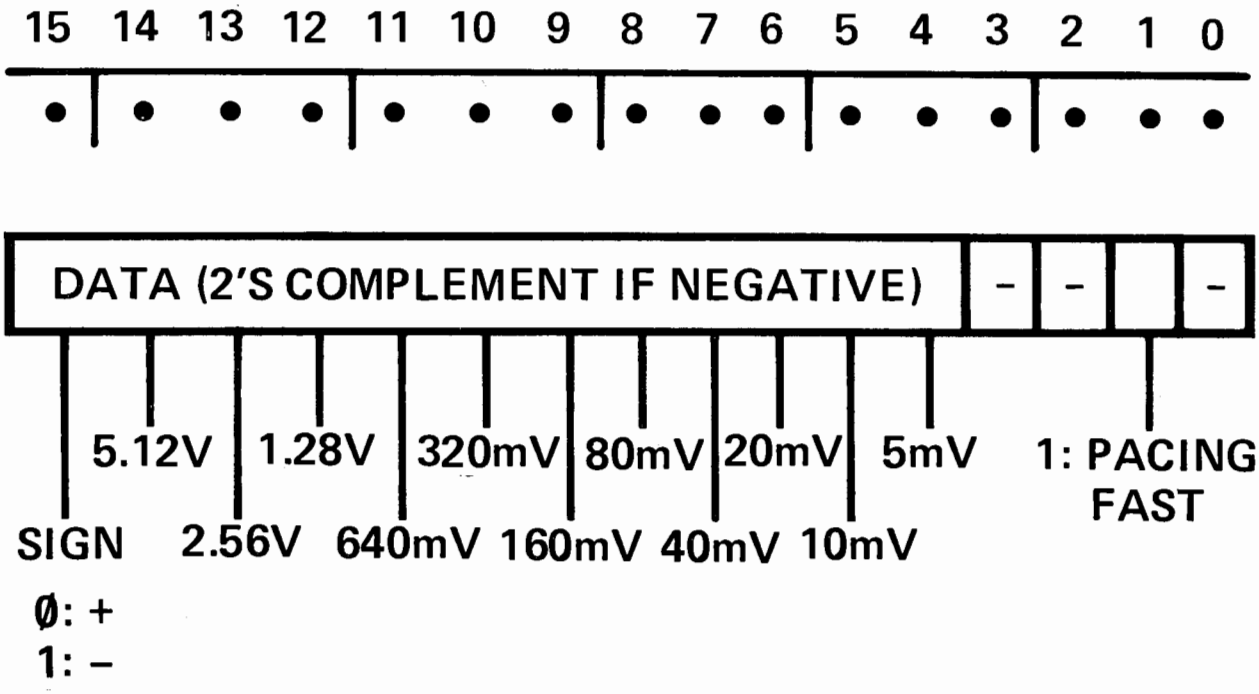
pace period = irate $*10^{(mult)}$

mode = start/stop control

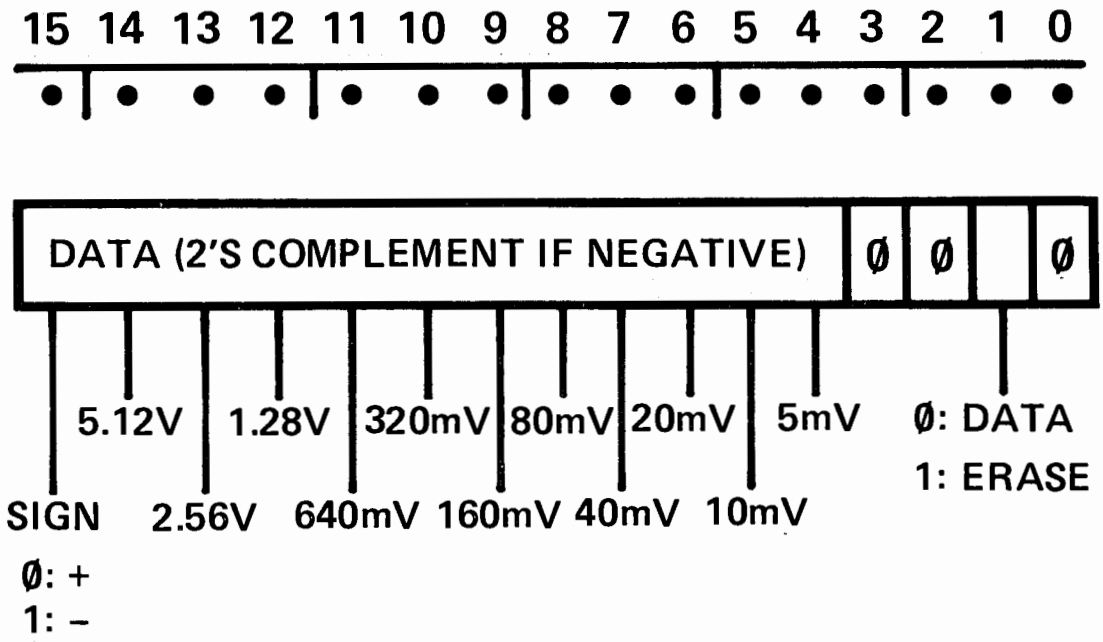
iunit = logical unit number of 2313B

ierr = error return parameter

SUBSYSTEM DATA RESPONSE WORD FORMAT



DUAL DAC DATA WORD



DIGITAL INPUT

CALL DI (num, ichan, idata, ierr)

num = number of words to be read

ichan = integer channel number array

idata = integer array for input data

ierr = error return parameter

MOMENTARY DIGITAL OUTPUT

CALL DOM (num, ichan, jdata, mult, ierr)

num = number of words to be written

ichan = integer channel number array

jdata = integer array for output data

mult = length of time for output to remain set (10's of msec)

ierr = error return parameter

LATCHED DIGITAL OUTPUT

CALL DOL (num, ichan, jdata, mask, ierr)

num = number of words to be written

ichan = integer array of channel numbers

jdata = integer array of output data

mask = integer array of output mask words

ierr = error return parameter

EVENT SENSING

CALL EVSNS (ichan, ibitn, ibit, iprog, ierr)

ichan = event sense channel number

ibitn = bit number to be addressed

ibit = value of bit ("0" or "1")

iprog = first two ASCII characters of the name of program to be scheduled, followed by three X's

ierr = error return parameter

CALL MPNRM

Erases previous bit-to-program relationships

HEWLETT hp PACKARD

22999-90026

PRINTED IN USA