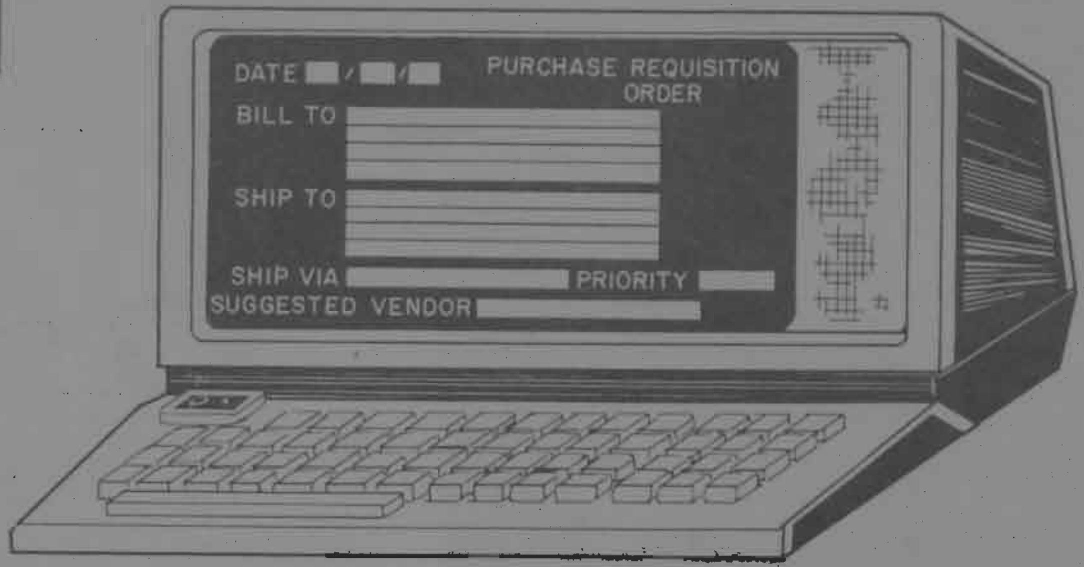
HEWLETT-PACKARD 3000

A COMPREHENSIVE INTRODUCTION

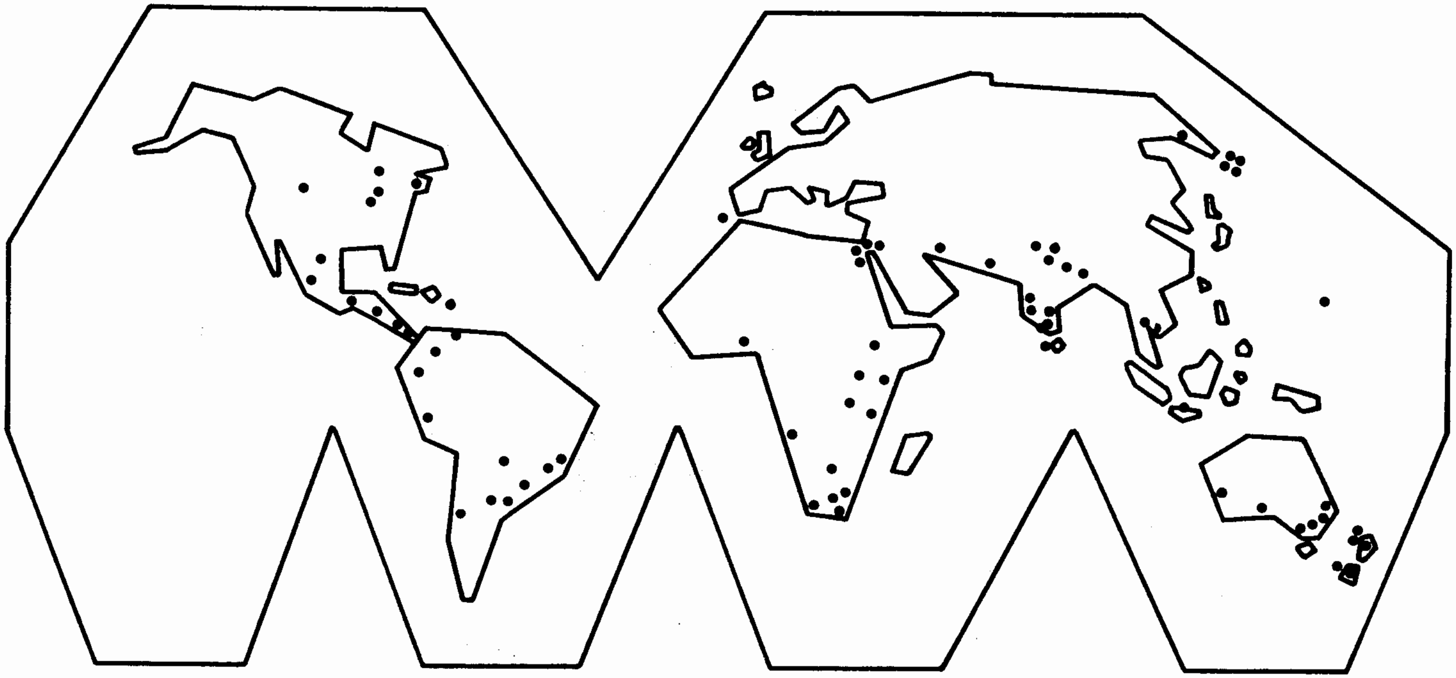


ORDER NUMBERS:

WORKBOOK	22801-90001
SLIDE SET	22801-90002
PRINT MASTERS	22801-90003

MAY 1976

HEWLETT PACKARD



- FOUNDED IN 1939 – ELECTRONIC INSTRUMENTATION
- 1966 – ENTERED COMPUTER INDUSTRY
- 1972 – HP 3000

30,000 EMPLOYEES IN 25 U.S. DIVISIONS
8 OVERSEAS MANUFACTURING PLANTS
172 SALES AND SERVICE FACILITIES IN 65 COUNTRIES

GENERAL SYSTEMS DIVISION
SANTA CLARA, CALIFORNIA

FORMED IN NOVEMBER, 1975 TO DEVELOP AND SUPPORT

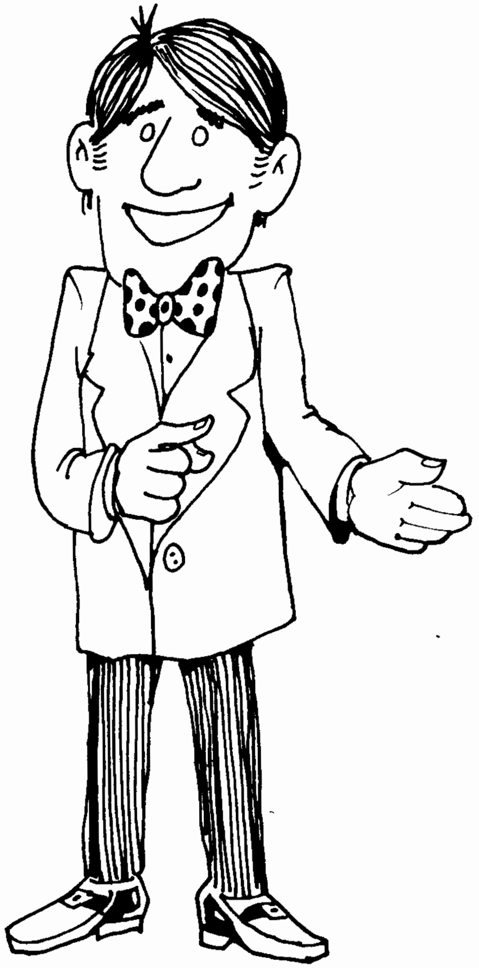
- HP 3000
- HP 2000 ACCESS

OTHER COMPUTER DIVISIONS

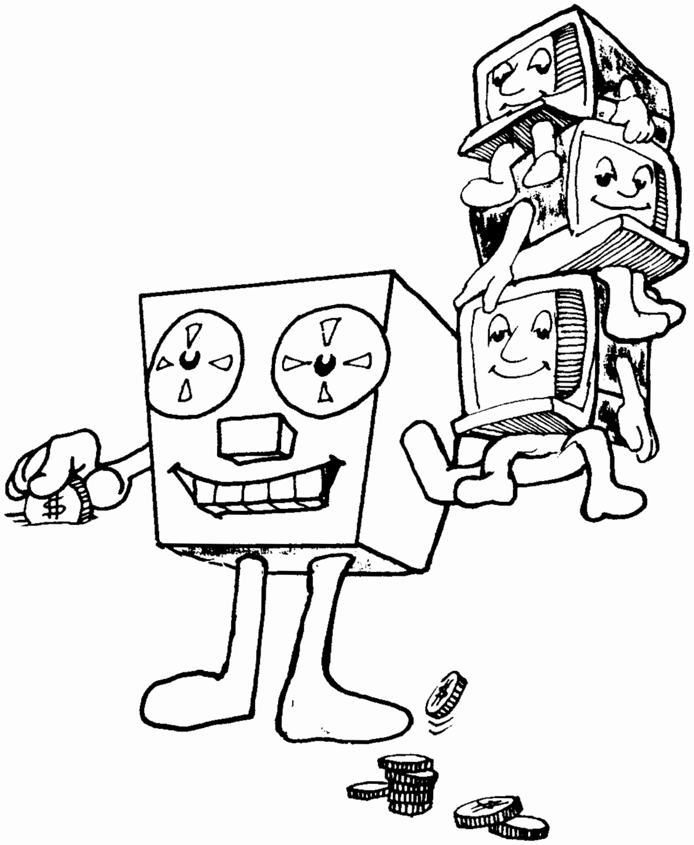
- AUTOMATIC MEASUREMENT SUNNYVALE, CA
  COMPUTER BASED AUTOMATIC TEST SYSTEMS
- DATA SYSTEMS CUPERTINO, CA
  MINICOMPUTER COMPONENTS
- DATA TERMINALS CUPERTINO, CA
  CRT'S AND RELATED PRODUCTS
- BOISE DIVISION BOISE, IDAHO
  PERIPHERALS
- HEWLETT PACKARD, FRANCE GRENOBLE, FRANCE
  COMPUTER PRODUCTION FOR EUROPEAN MARKET

COURSE SUMMARY

- SYSTEM OVERVIEW
- USER INTERFACE
- EDITOR
- FILE SYSTEM
- UTILITIES
- IMAGE/QUERY OVERVIEW
- LANGUAGES
- SEGMENTATION CONCEPTS
- DATA ENTRY LIBRARY



THE HEWLETT-PACKARD 3000 SERIES II

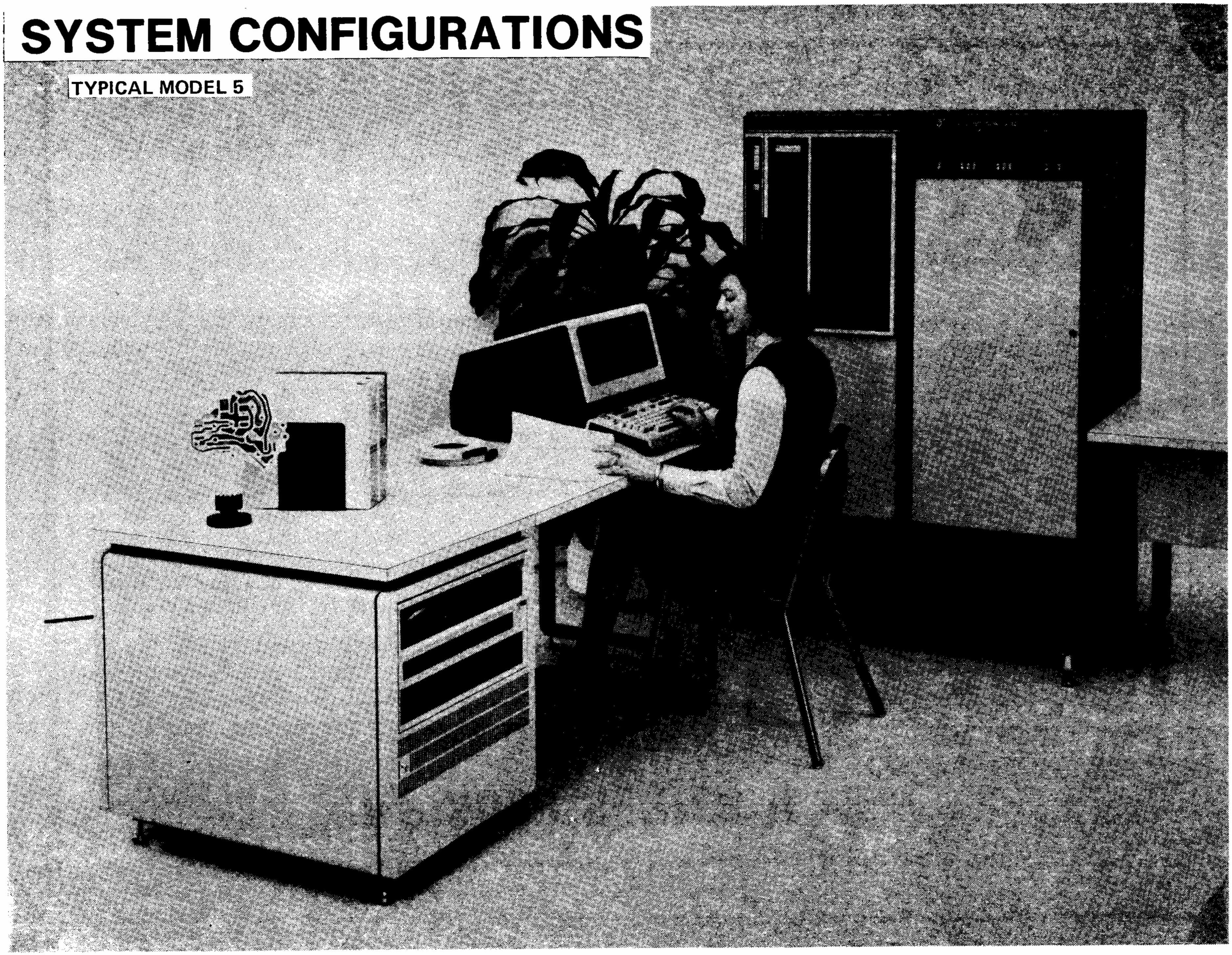


A NEW COST PERFORMANCE STANDARD

HP 3000 SERIES II CAPABILITIES

- CONCURRENT PROCESSING
- APPLICATION PROGRAM DEVELOPMENT
- HIGH DATA THROUGHPUT
- TERMINAL CAPABILITIES
- MULTIPROGRAMMING
- REMOTE JOB ENTRY

SYSTEM CONFIGURATIONS



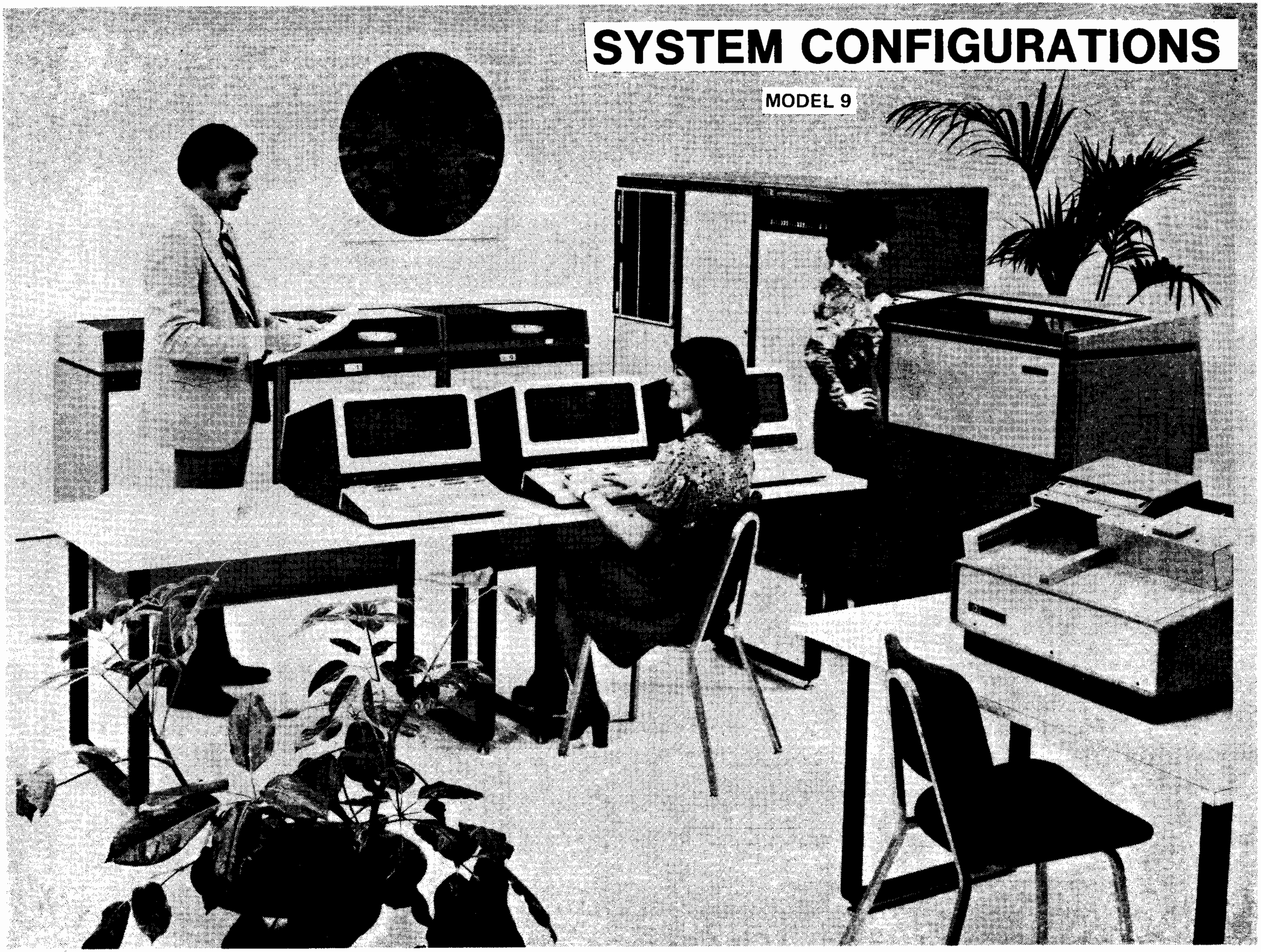
TYPICAL MODEL 5

SYSTEM CONFIGURATION

MODEL 7

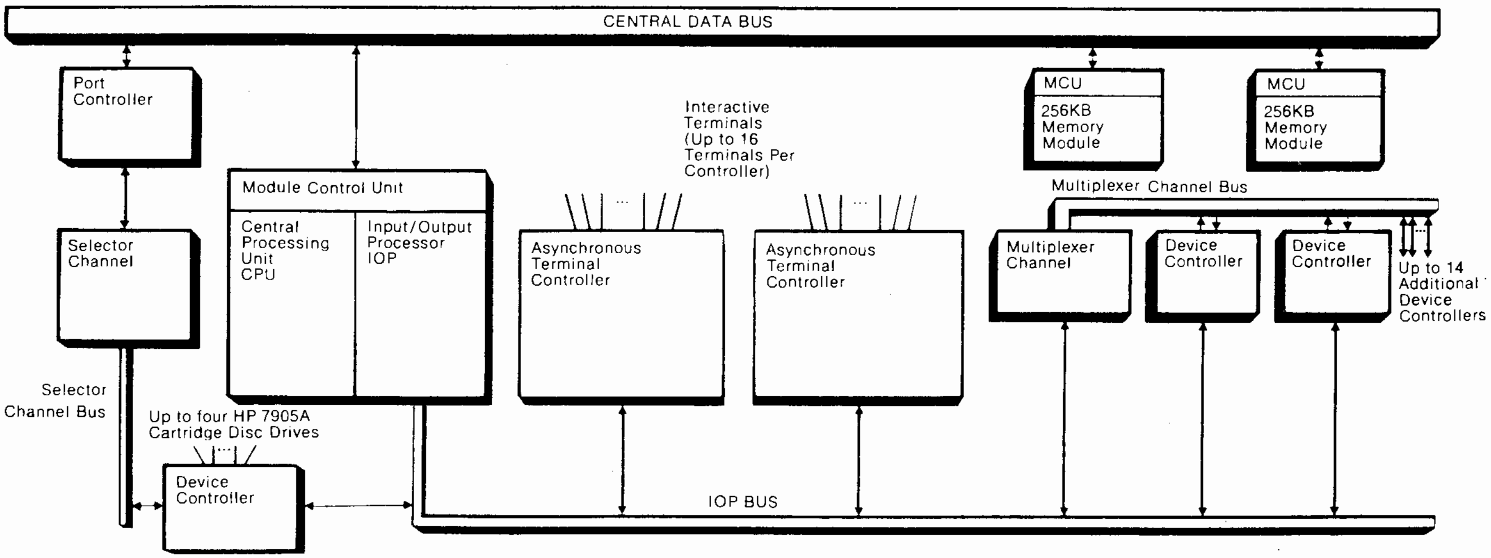
SYSTEM CONFIGURATIONS

MODEL 9



HP 3000 HARDWARE

Series II System Architecture


CENTRAL DATA BUS
Port Controller
Selector Channel
Selector Channel Bus
Module Control Unit
Central Processing Unit CPU
Input/Output Processor IOP
Up to four HP 7905A Cartridge Disc Drives
Device Controller
Interactive Terminals (Up to 16 Terminals Per Controller)
Asynchronous Terminal Controller
Asynchronous Terminal Controller
MCU
256KB Memory Module
MCU
256KB Memory Module
Multiplexer Channel Bus
Multiplexer Channel
Device Controller
Device Controller
Up to 14 Additional Device Controllers
IOP BUS

- CENTRAL PROCESSING UNIT
- MICROPROGRAMMED OPERATION
- INPUT/OUTPUT PROCESSOR
- CENTRAL DATA BUS
- FAULT CONTROL MEMORY

- MULTIPLEXOR CHANNEL
- SELECTOR CHANNEL
- ASYNCHRONOUS TERMINAL CONTROLLER
- STACK ARCHITECTURE

THE 3000 PROCESS

- A PROCESS IS THE BASIC "RUNNABLE" ENTITY MANAGED BY THE 3000 MULTI-PROGRAMMING SYSTEM

- A PROCESS MUST CONTAIN AT LEAST:

  1. ONE GROUP OF INSTRUCTIONS CALLED:

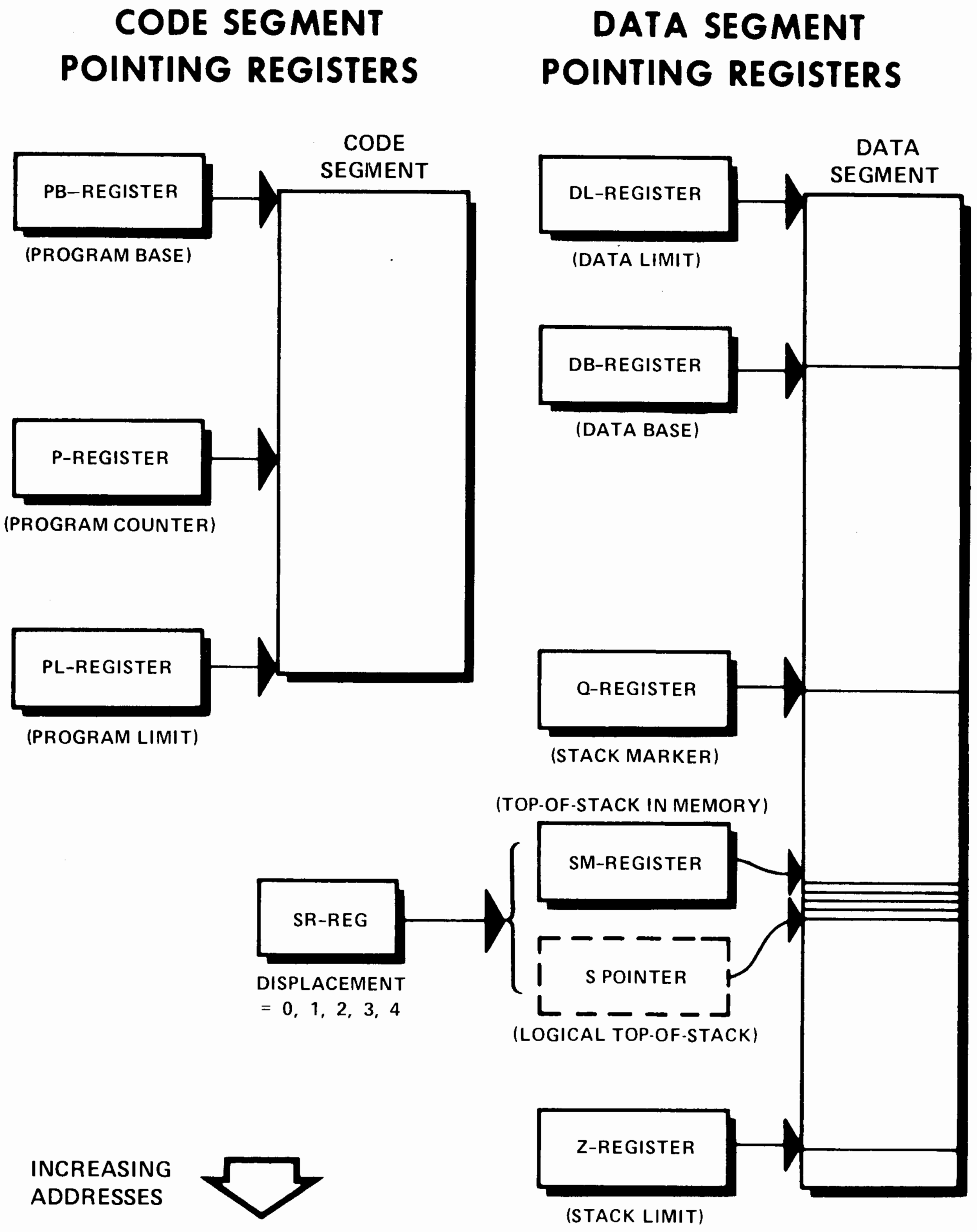
     —A CODE SEGMENT—

  2. A DATA DOMAIN THAT IS ABSOLUTELY PRIVATE CALLED:

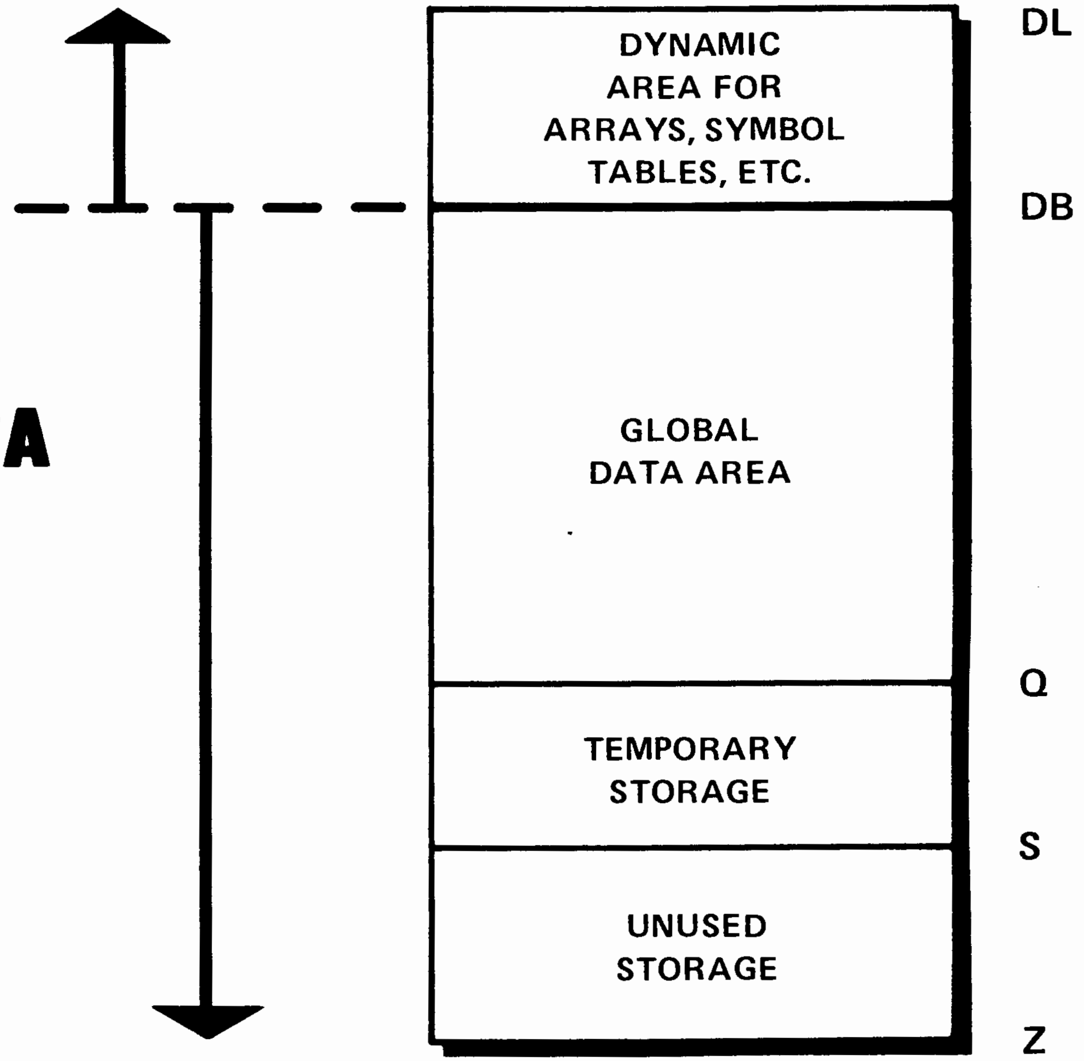
     THE STACK

- A PROCESS HAS A UNIQUE STACK, BUT MAY HAVE MORE THAN ONE CODE SEGMENT.

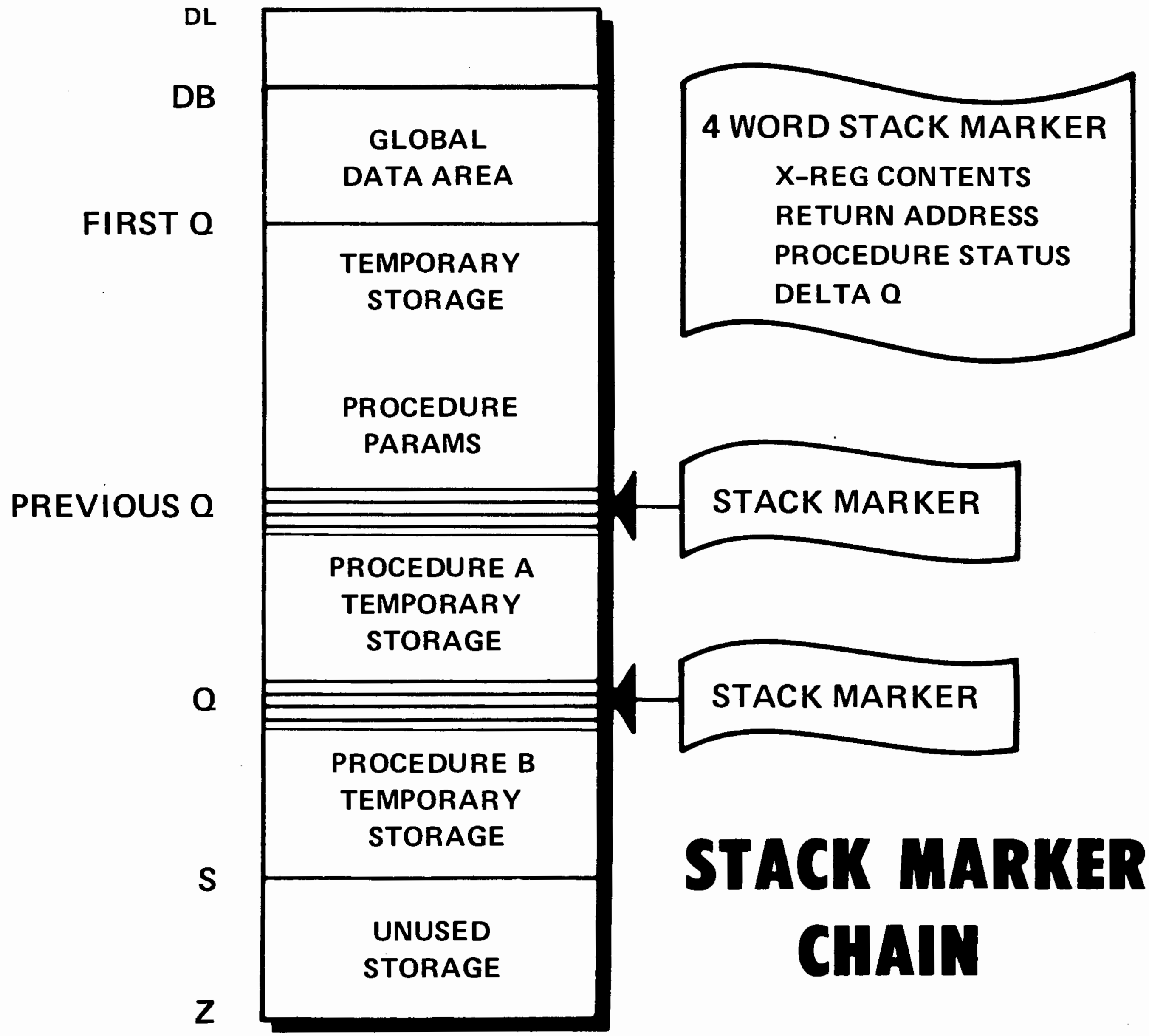
- A PROCESS IS THE FLOW OF CONTROL THROUGH PROGRAM CODE


CODE SEGMENT
POINTING REGISTERS
DATA SEGMENT
POINTING REGISTERS
CODE
SEGMENT
PB–REGISTER
(PROGRAM BASE)
P-REGISTER
(PROGRAM COUNTER)
PL-REGISTER
(PROGRAM LIMIT)
DATA
SEGMENT
DL-REGISTER
(DATA LIMIT)
DB-REGISTER
(DATA BASE)
Q-REGISTER
(STACK MARKER)
(TOP-OF-STACK IN MEMORY)
SM-REGISTER
SR-REG
DISPLACEMENT
= 0, 1, 2, 3, 4
S POINTER
(LOGICAL TOP-OF-STACK)
Z-REGISTER
(STACK LIMIT)
INCREASING
ADDRESSES

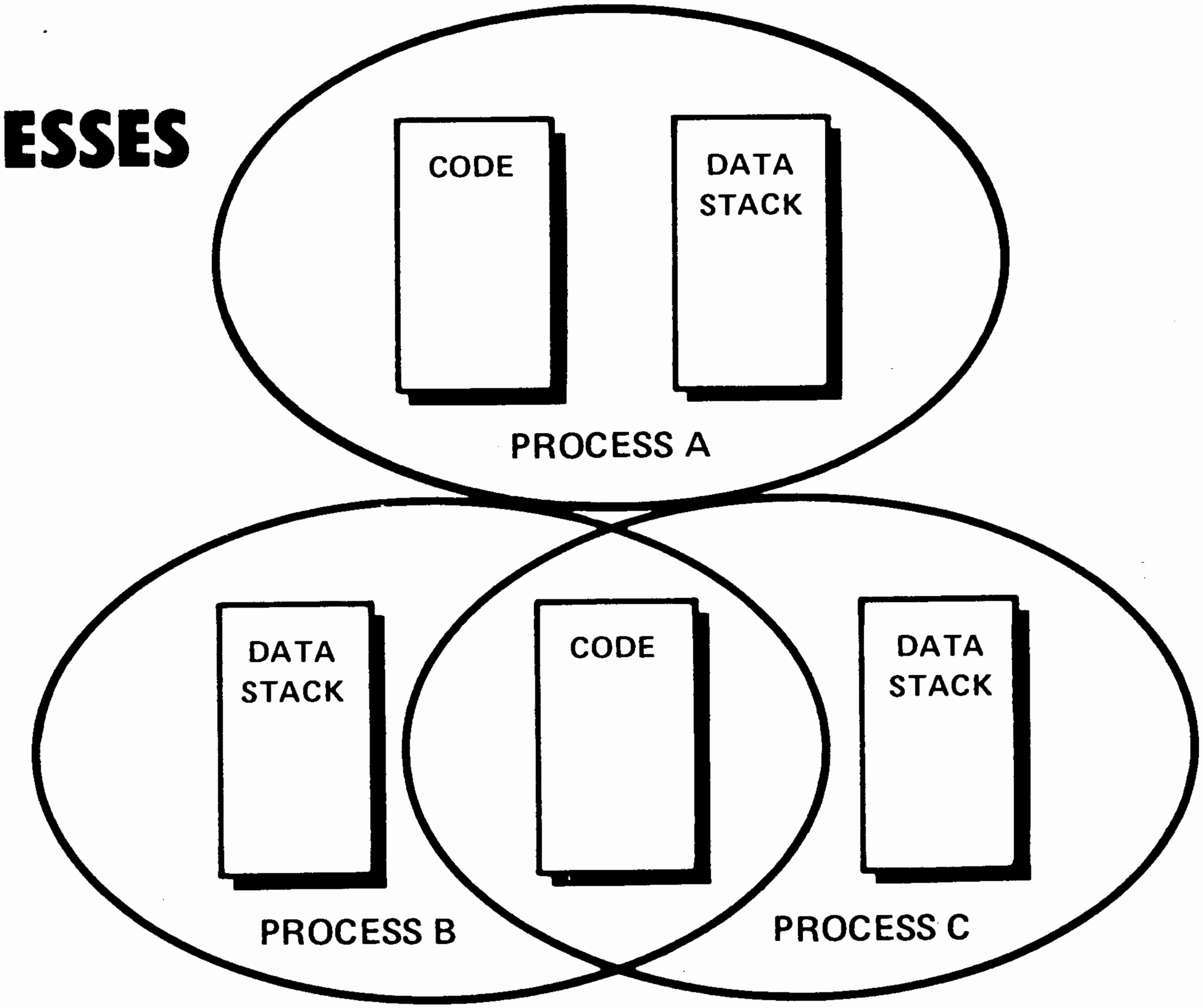
STACK DATA STORAGE



STACK MARKER CHAIN

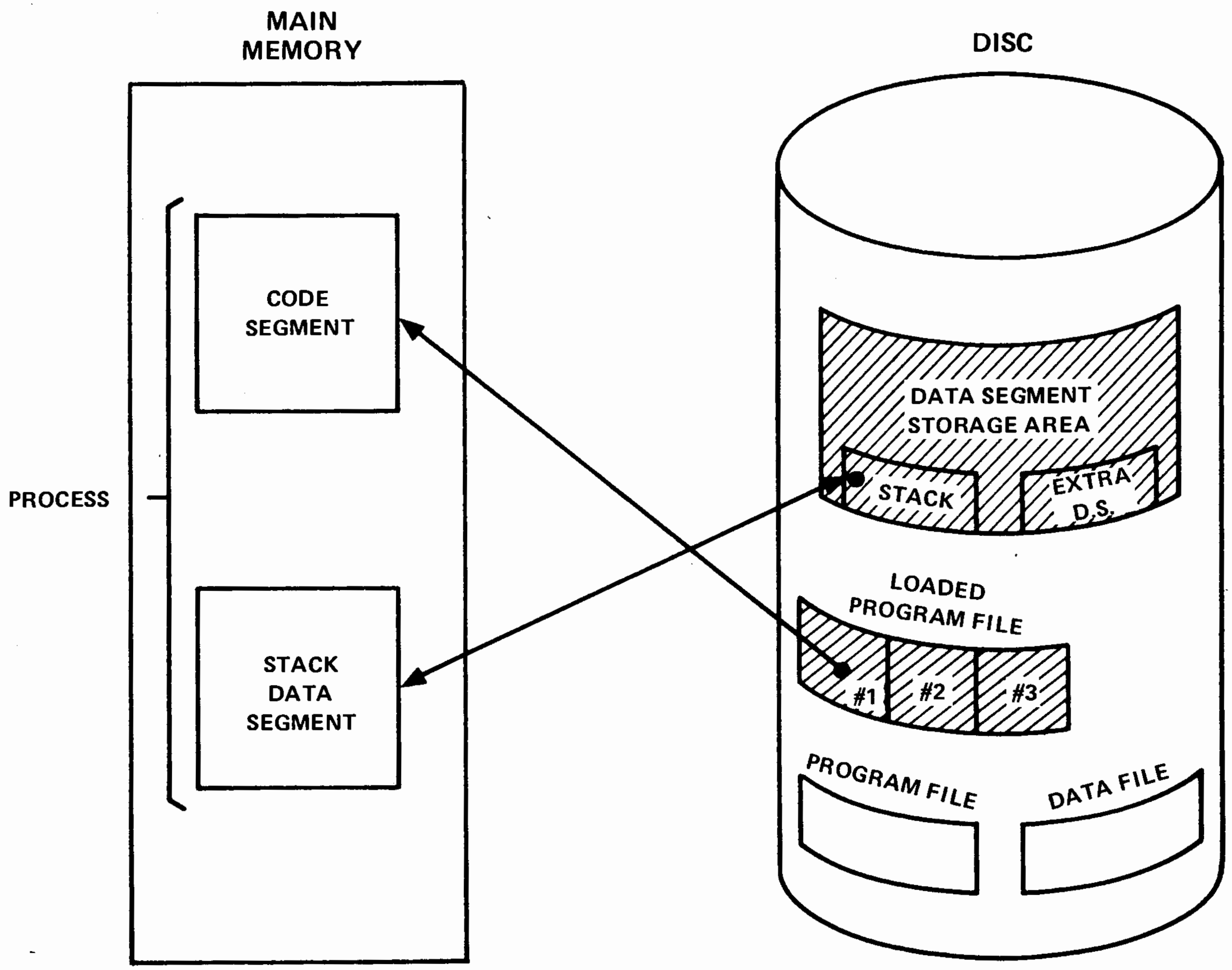


CODE SHARING BY PROCESSES



STACK ARCHITECTURE BENEFITS

- USER/SYSTEM PROTECTION
- REDUCED DISC TRANSFERS
- CODE COMPRESSION
- EFFICIENT EXPRESSION EVALUATION
- POWERFUL PROCEDURE PROCESSING
- DYNAMIC STORAGE ALLOCATION



VIRTUAL MEMORY CONCEPT

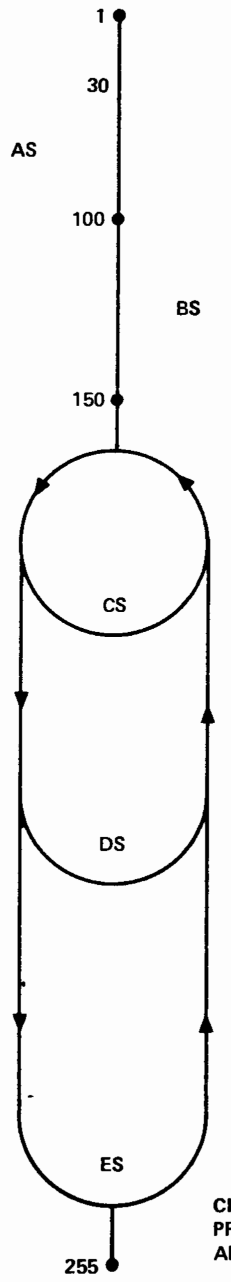
▨ VIRTUAL MEMORY SHADED

HP 3000 SERIES II

- SEGMENT FAULT FREQUENCY ALGORITHM
- WORKING SET CONCEPT
- LOCAL COMPRESSION ALGORITHM

HP 3000 SERIES II

MASTER QUEUE STRUCTURE



CIRCULAR PROCESSES FALL IN
PRIORITY AS THEY GET SERVICE,
AND RISE IN PRIORITY AS THEY WAIT.

- BATCH/INTERACTIVE PROCESSING
- USER-ORIENTED JCL
- ADVANCED FILE SYSTEM
- COMPREHENSIVE FILE SECURITY
- AUTOMATIC SPOOLING
- SIX HOST LANGUAGES
- DATA BASE MANAGEMENT
- UTILITY SUBSYSTEMS
- AUTO SYSTEM LOGGING
- HP SYSTEMS COMPATABILITY

SOFTWARE FEATURES

LAB # 1 INTRODUCTION

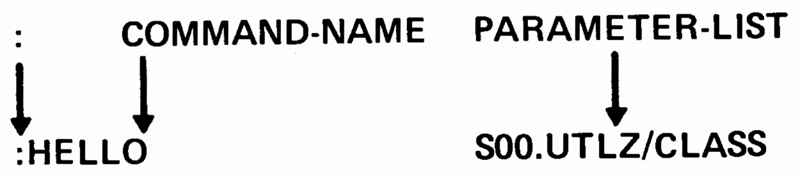
1. Obtain your user identification from the Instructor. This will be your user and account name for the duration of the course.

2. Complete Section 1 of Using the HP3000, A Guide for the Terminal User, on the system.

COMMUNICATING WITH MPE

- COMMANDS
- PROGRAMMATICALLY

MPE/3000 COMMANDS



```
:      COMMAND-NAME   PARAMETER-LIST
:HELLO                S00.UTLZ/CLASS
```

1. ENTIRE RECORD IS USED.

2. THERE MUST BE A BLANK (OR PUNCTUATION) BETWEEN COMMAND-NAME AND PARAMETER-LIST.

3. PARAMETERS ARE POSITIONAL (,) OR KEYWORD (;).

4. ALL NUMBERS DECIMAL, UNLESS STARTED WITH %.

5. & AS LAST NON-BLANK, CAUSES CONTINUATION.

POSITIONAL PARAMETERS

- SEPARATED BY COMMAS

  COMMAS USED AS PLACE-HOLDER.
  POSITION IN LIST DETERMINES MEANING.
  DEFAULT IS USED WHEN THEY ARE LEFT OUT.

- EXAMPLES

```
:RPG TEXT, USL, LIST
:RPG  ,,LIST
:RPG TEXT
```

KEYWORD PARAMETERS

- PRECEDED BY SEMICOLON
- CAN APPEAR IN ANY ORDER (AFTER POSITIONALS) SINCE "KEY WORD" INDICATES MEANING
- CAN HAVE POSITIONAL SUBPARAMETER – LIST

```
EXAMPLES      :FILE T1; TEMP
              :FILE TAP;DEV=TAPE
              :FILE A;REC=–80,10; DISC=20,4,2
```

DOCUMENTATION CONVENTIONS

- UPPER CASE FOR LITERAL WORDS
- LOWER CASE FOR VARIABLE ITEMS
- [BRACKETS] FOR OPTIONAL ITEM
  (DEFAULT ALTERNATIVE MAY BE UNDERLINED)
- {BRACES} REQUIRED ITEM, CHOOSE ONE OF SEVERAL

MPE 3000 COMMAND SYNTAX WORKSESSION

In these exercises, determine whether examples shown are valid according to rule above them.

1. :RESET $\begin{Bmatrix} \text{formaldesignator} \\ \text{@} \end{Bmatrix}$

	OK	BAD
:RESET		
:RESET formaldesignator		
:RESET @, formaldesignator		
:RESET @		

2. :COBOLGO [sourcefile] [,[listfile]
[,[master file] [,new file]]]

Assume that T4 and LLM are source files,
*LP is a listfile,
VERSION1 is a masterfile,
VERSION2 is a newfile.

	OK	BAD
:COBOLGO T4,,*LP		
:COBOLGO		
:COBOLGO T4,*LP,,VERSION2		
:COBOLGO LLM,,VERSION1, VERSION2		
:COBOLGO ,,,VERSION2		
:COBOLGO LLM,*LP		
:COBOLGO T4, VERSION1, VERSION2		

WORKSESSION (CONT'D)

3. :HELLO username [/password] . acctname [/password]
              [,groupname [/password]]
              [;TERM=termtype]
              [;TIME=cpu time limit]

	OK	BAD
:HELLO A/C2.B; TIME=3		
:HELLO A/C2,B/C3		
:HELLO MANAGER.SYS, TERM=5		
:HELLO S00.UTLZ,G00		
:HELLO A/C2.B/C3,C/C4 & :;TIME=3; TERM=4		
:HELLO USER.ACCOUNT–3; TIME=5		
:HELLO		
:HELLO USER		

worksession (cont'd)

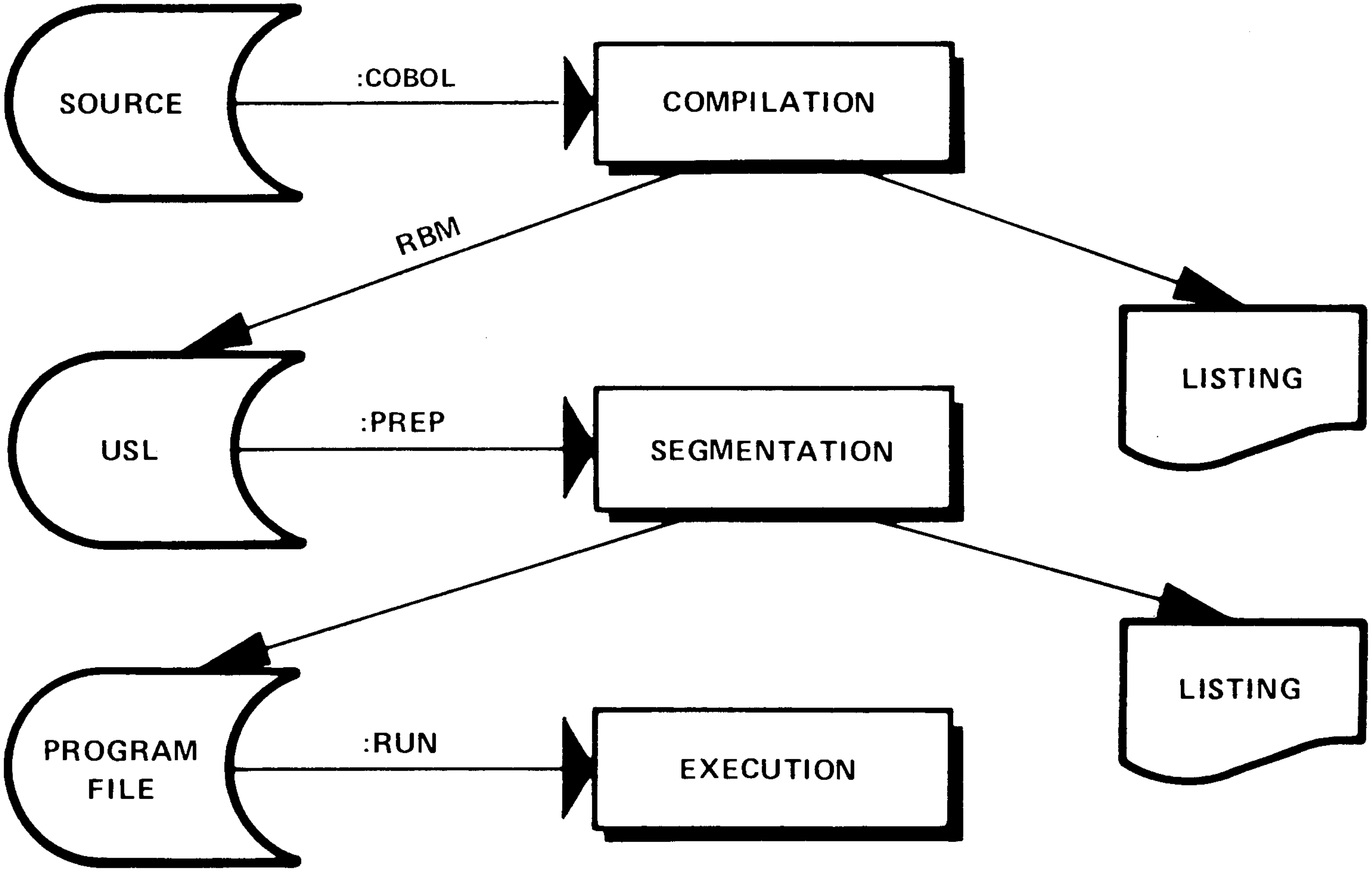
```
4.   :FILE formaldesignator  ┌=$ NEWPASS                              ┐
                             │[= filereference]  [,NEW]               │
                             │                                        │
                             │ =$OLDPASS           ┌              ┐   │
                             │                     │,OLD          │   │
                             └[=filereference]     └,OLDTEMP      ┘   ┘

     [;REC = [recsize] [, [blockfactor] [, [F] [,BINARY] ] ] ] ] *
                                           [U] [,ASCII ]
                                           [V]

     [;CCTL  ]*
     [;NOCCTL]

     [           ( IN       ) ]
     [           ( OUT      ) ]
     [ ;ACC  =  { UPDATE    } ]
     [           ( OUTKEEP  ) ]
     [           ( APPEND   ) ]
     [           ( INOUT    ) ]

     [;NOBUF                ]
     [;BUF   [=numbuffers]  ]

     [;EXC]
     [;EAR]
     [;SHR]

     [;MR  ]
     [;NOMR]

     [;DEL ]
     [;SAVE]
     [;TEMP]

     [;DEV = [device] [, [outputpriority] [,numcopies] ] ] *

     [;CODE = filecode] *

     [;DISC = [filesize] [,[numextents] [, initalloc] ] ] *

     [;NOWAIT]
     [;WAIT  ]

     [;MULTI  ]
     [;NOMULTI]

   * do not apply to old disc files
```

WORKSESSION (CONT'D)

	OK	BAD
:FILE TAPE; DEV = TAPE		
:FILE TEMP; TEMP		
:FILE S26 = TEMP, EXC, NOBUF		
:FILE LP, DEV = LP; REC = –80		
:FILE ;REC = 40,1,,BINARY; NOMR		
:FILE A; CCTL; ACC = NOBUF, EAR, MR		
:FILE T; DISC = ,4; SAVE		
:FILE ABCDEFGHIJ = $NEWPASS		
:FILE 2ND; CODE = 1029		
:FILE T; DEV = TAPE; :& REC = –80, 10, V, BINARY		
:FILE A = T, NEW; DISC = 15, 1,1;& :REC = 40,3,F,ASCII; DEL; EXC; & :BUF = 4; NOCCTL		
:FILE T; & :FILE DEV = TAPE		

MPE INTRINSICS

- SYSTEM PROCEDURES WHICH ALLOW THE USER TO PERFORM CERTAIN SYSTEM FUNCTIONS PROGRAMMATICALLY
- WRITTEN IN SPL AND FOLLOW THE RULES AND CONSTRAINTS OF THAT LANGUAGE
- CAN BE CALLED DIRECTLY FROM ANY SPL, FORTRAN PROGRAM



MPE PROGRAM MANAGEMENT

PROGRAM MANAGEMENT UNDER

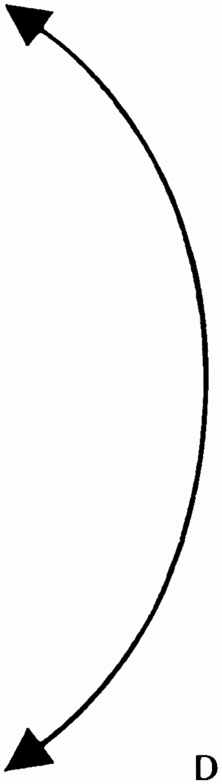
MPE 3000

COMPILATION: SOURCE CODE IS COMPILED INTO OBJECT MODULES CALLED RELOCATABLE BINARY MODULES (RBM'S). THESE ARE STORED IN A SPECIAL FILE CALLED A USER SUBPROGRAM LIBRARY (USL)

PREPARATION: SEGMENTER SUBSYSTEM PREPARES THE RBM'S INTO SEGMENTS AND STORES THEM IN A PROGRAM FILE

EXECUTION: ENTRIES MADE IN APPROPRIATE SYSTEM TABLES; DATA SEGMENT OBTAINED; CODE SEGMENTS ALLOCATED.

BATCH MODE OPERATION



INPUT THROUGH SERIAL DEVICE

INITIATED BY DEVICE INTERRUPT

USER SUPPLIES COLON

COMMAND ERROR MAY CAUSE JOB ABORT

COMMANDS AND DATA CAN BE INTERMIXED

DELIMITED BY "JOB" AND "EOJ"

MPE "JOB CONTROL" COMMANDS

:JOB [jobname,] username [/upass] .acct name [/apass] [,groupname/gpass]

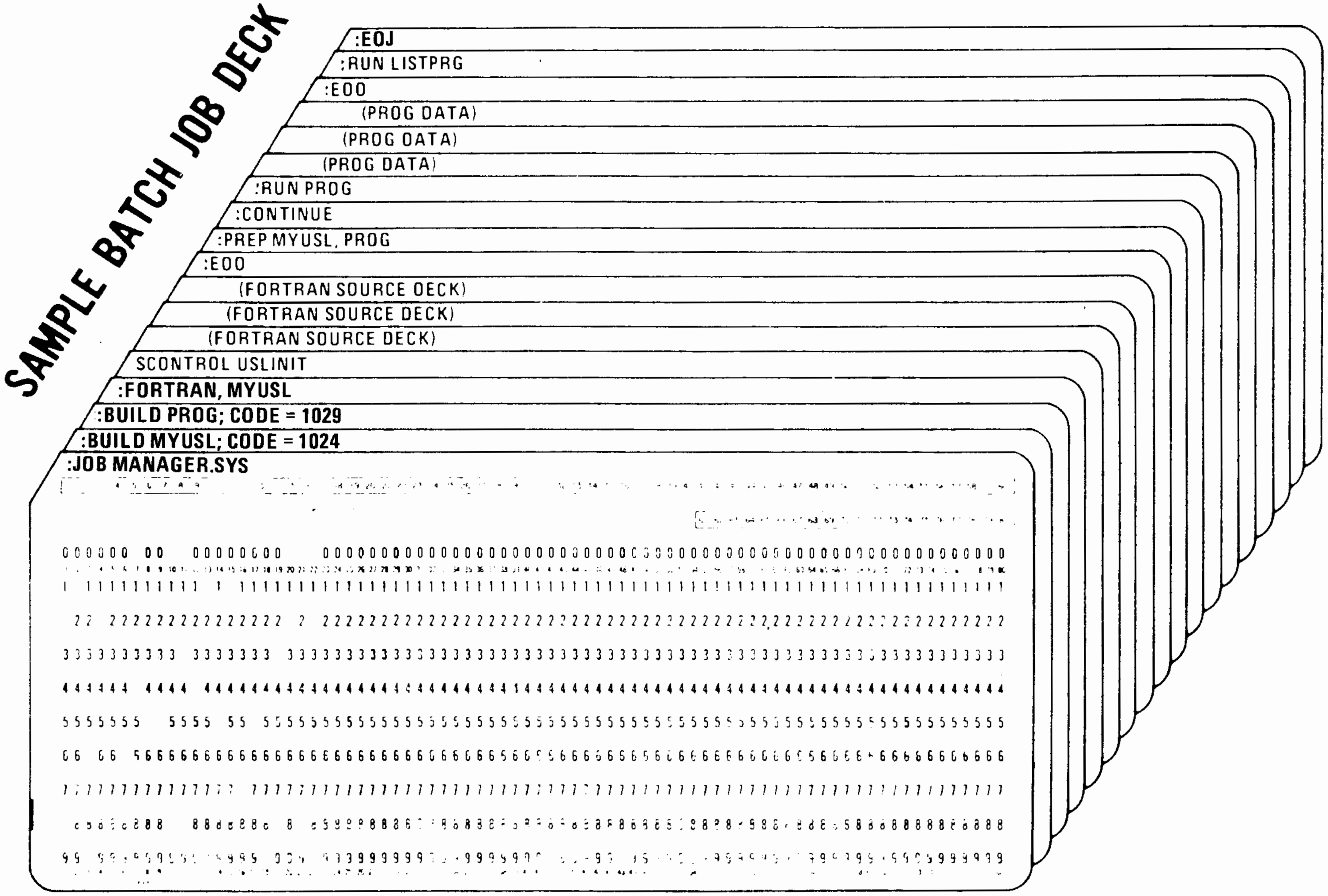
[;TIME = cputime] [;PRI = execute priority] [;HIPRI
;INPRI = input priority]

[;OUTCLASS = [device] [,outpriority] [,numcopies]] [;RESTART]

```
:JOB TX78,NEOPHYTE/IFORGOTT.COMMERCE/WHATISIT,PROJECT3/SECRET33  &
:    ;TIME=6   ;PRI=CS   ;INPRI=8   ;OUTCLASS=,10,3   ;RESTART

:EOJ
```

```
:JOB    TX78,NEOPHYTE.COMMERCE,PROJECT3
  PRI=CS;   INPRI=8    ;TIME=6
 JOB NUMBER= #J11
 TUE, MAR   4, 1975   8:17 AM
 HP32000C.00.01


:EOJ

 CPU (SEC) = 1
 ELAPSED (MIN) = 1
 TUE, MAR   4, 1975  8:17 AM
 END OF JOB
```

:EOJ


SAMPLE BATCH JOB DECK
:EOJ
:RUN LISTPRG
:EOD
(PROG DATA)
(PROG DATA)
(PROG DATA)
:RUN PROG
:CONTINUE
:PREP MYUSL, PROG
:EOD
(FORTRAN SOURCE DECK)
(FORTRAN SOURCE DECK)
(FORTRAN SOURCE DECK)
SCONTROL USLINIT
:FORTRAN, MYUSL
:BUILD PROG; CODE = 1029
:BUILD MYUSL; CODE = 1024
:JOB MANAGER.SYS

SESSION MODE OPERATION

INPUT THROUGH INTERACTIVE TERMINAL

INITIATED BY RETURN KEY

SYSTEM SUPPLIES COLON

NO SESSION ABORT ON COMMAND ERROR

COMMANDS AND DATA CAN BE INTERMIXED

PROGRAMS CAN BE INTERRUPTED BY BREAK KEY

DELIMITED BY "HELLO" AND "BYE"

HELLO COMMAND PARAMETERS

- ▷ SESSION NAME
  - ▶ USER NAME (REQ)
    - ▶ "PERIOD" ACCOUNT NAME (REQ)
  - ▷ "COMMA" GROUP NAME
  - ▷ TERMINAL TYPE
- ▷ PRIORITY

```
:HELLO FRED,FINANCE.INFOSYS;PRI=DS
```

HELLO COMMAND

:HELLO [sessionname,] username[/upass].*acctname* [/apass] [,groupname/gpass]

[;TERM = termtype] [;TIME = cputime] [;PRI = executepriority]

[;HIPRI
;INPRI = inputpriority]

```
:HELLO TX78,NEOPHYTE/IFORGOTT.COMMERCE/WHATISIT,PROJECT3/SECRET33 &
:       ;TERM=4    ;TIME=6     ;PRI=CS   ;INPRI=13
 SESSION NUMBER = #S221
 THU, FEB 27, 1975,  1:11 PM
 HP32000C.ØØ.Ø1

:BYE

CPU (SEC) = 1
CONNECT (MIN) = 1
THU, FEB 27, 1975,  1:12 PM
END OF SESSION
```

```
:HELLO NEOPHYTE.COMMERCE,PROJECT3
ACCT  PASSWORD?
USER  PASSWORD?
GROUP PASSWORD?
 SESSION NUMBER = #S230
 THU, FEB 27, 1975,  1:23 PM
 HP32000C.ØØ.Ø1

:BYE
```

```
:HELLO BOB.JOHNSON
ACCT  PASSWORD?
 SESSION NUMBER = #S232
 THU, FEB 27, 1975,  1:24 PM
 HP32000C.ØØ.Ø1

:BYE
```

:BYE

CONTROL CHARACTERS DURING SESSION

CONTROL H (H^C)

HOLD DOWN CONTROL WHILE YOU STRIKE H.

DELETES LAST CHARACTER IN BUFFER.

CONTROL X (X^C)

DELETES CURRENT LINE.

PRINTS THREE EXCLAMATION POINTS (! ! !), RETURN, LINE FEED.

NO NEW PROMPT CHARACTER.

CONTROL Y (Y^C)

USED BY SOME SUBSYSTEMS TO BREAK INPUT, ETC.

<u>NOT USED</u> IN MPE COMMAND MODE.

BREAK

USED TO SUSPEND SUBSYSTEMS AND ENTER MPE.

<u>NOT USED</u> TO "BREAK" MPE OPERATIONS.

CONTROL CHARACTERS (cont'd)

CONTROL F (F^{C})

RECOVER FROM TERM TYPE (10) ERROR

CONTROL Q (Q^{C})

PLACE TERMINAL IN TAPE MODE
(CNTRL Y TO STOP)

ESC;

STOP ECHOING

ESC:

RESUME ECHOING

MISCELLANEOUS "MPE" COMMANDS

```
:RUN  programfilename

  :ABORT

     :COMMENT  text

        :CONTINUE

           :PTAPE  [filename]

              :REPORT

                 :RESUME

                           ⎧ jsnumber          ⎫
                    :TELL  ⎨          ;message ⎬
                           ⎩ jsname            ⎭
                             :TELLOP  message
```

MPE "JOB CONTROL" COMMANDS

:SETMSG $\left\{\begin{array}{l}\underline{\text{ON}}\\ \underline{\text{OFF}}\end{array}\right\}$

:SHOWTIME

:SPEED $\left\{\begin{array}{l}\text{[inspeed] , }\underline{\text{outspeed}}\\ \underline{\text{inspeed}}\end{array}\right\}$

INFORMATION DISPLAY COMMANDS

:SHOWDEV [ℓdev | classname]

```
:SHOWDEV PHONE
LDEV  AVAIL       OWNERSHIP

 55 A AVAIL
 35 A AVAIL

:SHOWDEV 5
LDEV  AVAIL       OWNERSHIP

  5 A SPOOLED     SPOOLER IN
```

INFORMATION DISPLAY COMMANDS

```
                 ┌                     ┐
                 │SP                   │
                 │#INNN                │
:SHOWIN          │STATUS               │
                 │ITEM[;ITEM[;ITEM]]   │
                 └                     ┘
```

```
NOTE:
ITEMS:
[DEV =  LDEV]

┌        ┌@J   ┐ ┐
│        │@S   │ │
│JOB  =  │#JNNN│ │
│        │#SNNN│ │
└        └     ┘ ┘

┌ACTIVE┐
│READY │
└OPENED┘
```

DO NOT USE DUPLICATE ITEM KEYWORDS IN THIS COMMAND.

```
:SHOWIN

DEV/CL    DFID     JOBNUM   FNAME     STATE FRM SPACE RANK PRI #C

45        #I50     #S30     $STDIN    OPENED
5         #I51                        READY           8
                   MGR.COMUTLZ

2 FILES:
    0 ACTIVE
    1 READY; INCL 1 SPOOFLES, 0 DEFERRED
    1 OPENED; INCL 0 SPOOFLES
    1 SPOOFLES: 8 SECTORS
```

INFORMATION DISPLAY COMMANDS

$$:\text{SHOWOUT}\quad \left[\begin{array}{l}\text{SP}\\ \#\text{O}nnn\\ \text{STATUS}\\ \text{ITEM}[;\text{ITEM}[;\text{ITEM}]]\end{array}\right]$$

NOTE:

ITEMS:

$$\left[\text{DEV} = \left\{\begin{array}{l}\text{LDEV}\\ \text{CLASS}\end{array}\right\}\right]$$

$$\left[\text{JOB} = \left\{\begin{array}{l}@\text{J}\\ @\text{S}\\ \#\text{J}nnn\\ \#\text{S}nnn\end{array}\right\}\right]$$

$$\left[\begin{array}{l}\text{ACTIVE}\\ \text{READY}\\ \text{OPENED}\end{array}\quad \left[\begin{array}{l},\text{N}\\ ,\text{D}\end{array}\right]\right]$$

DO NOT USE DUPLICATE ITEM KEYWORDS IN THIS COMMAND.

```
:SHOWOUT

DEV/CL    DFID     JOBNUM   FNAME     STATE FRM SPACE RANK PRI #C

45        #061     #S30     $STDLIST  OPENED

OUTFENCE= Ø
```

INFORMATION DISPLAY COMMANDS

```
:SHOWJOB    [ #JNNN           ]
            [ #SNNN           ]
            [ STATUS          ]
            [ ID[;STATE]      ]
            [ STATE[;ID]      ]
            [ SUSP            ]

   NOTE:
   ID:
   [        { @J                      } ]
   [        { @S                      } ]
   [ JOB =  { [JSNAME,] USER.ACCT     } ]
   [        { @,USER.ACCT             } ]
   [        { [@,] @.ACCT             } ]

   STATE:
        [ INTRO        ]
        [      [,N]    ]      N = NON-DEFERRED
        [ WAIT [,D]    ]      D = DEFERRED
        [ EXEC         ]
        [ SUSP         ]
```

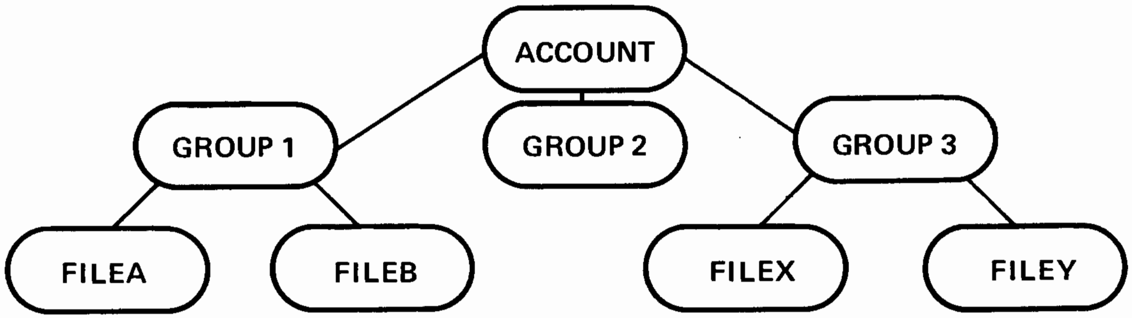
```
:SHOWJOB

JOBNUM  STATE IPRI JIN  JLIST    INTRODUCED  JOB NAME

#S25    EXEC        23   23      WED 10:57A  MGR.LIB
#S26    EXEC        45   45      WED 11:00A  MGR.COMUTLZ

2 JOBS:
    0 INTRO
    0 WAIT; INCL 0 DEFERRED
    2 EXEC; INCL 2 SESSIONS
    0 SUSP
JOBFENCE= 0; JLIMIT= 3; SLIMIT= 16
```

THE ACCOUNT

- MAJOR BILLABLE ENTITY FOR RESOURCES USED
- SYSTEM MANAGER DEFINES EACH ACCOUNT
  - NAME AND (PASSWORD)
  - ACCOUNT MANAGER
  - CAPABILITY LIST
  - SCHEDULING PRIORITY
  - MAXIMUM FILE SPACE, CPU TIME, CONNECT TIME
  - FILE ACCESS MODES
- SYSTEM KEEPS RUNNING TOTAL OF RESOURCE USAGE FOR EACH ACCOUNT

THE GROUP

- PROVIDES SUBDIVISION FOR FILES WITHIN AN ACCOUNT



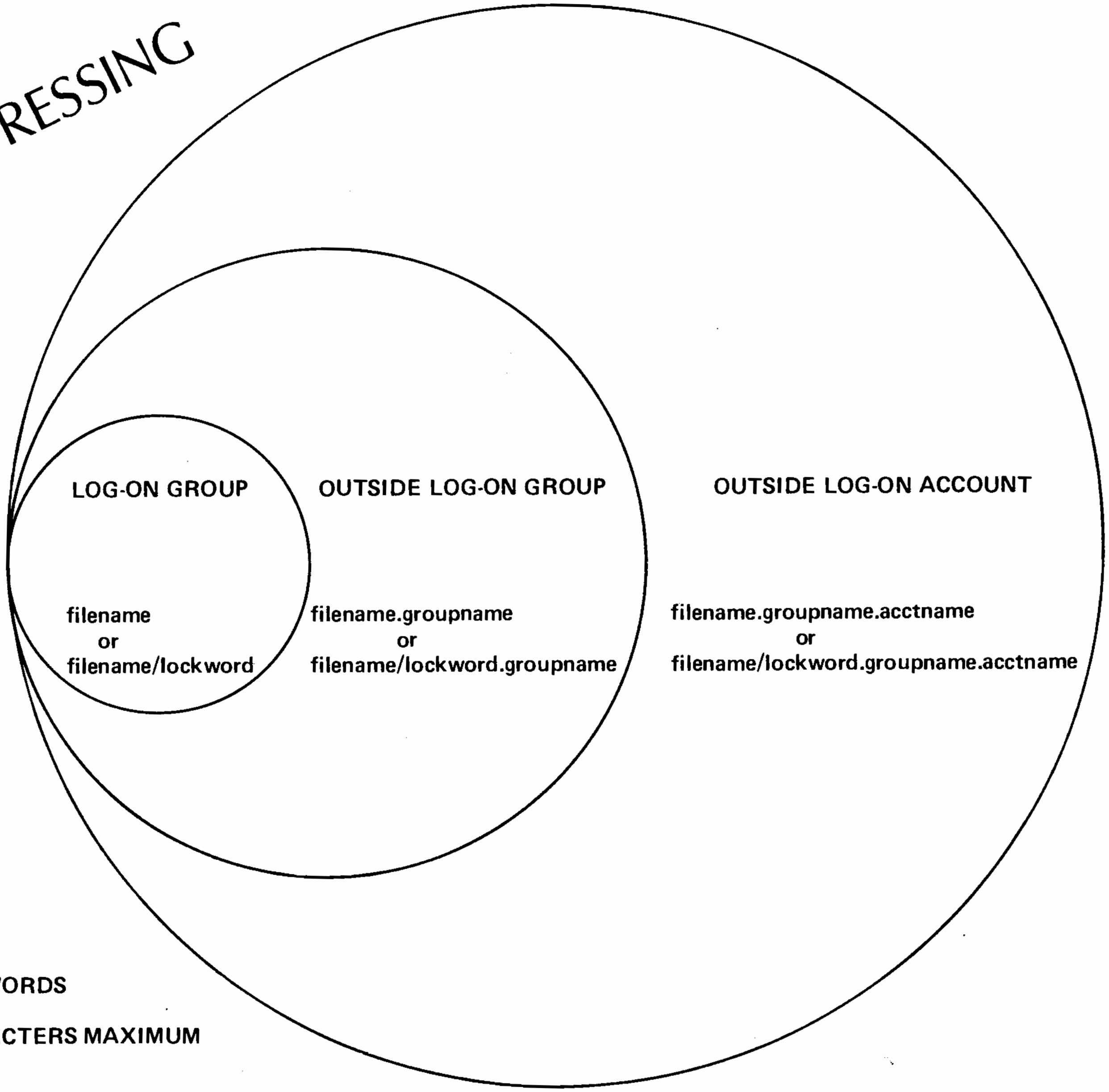
- CREATED BY THE ACCOUNT MANAGER
  - NAME AND (PASSWORD)
  - LIMITS ON FILE SPACE, CPU TIME, CONNECT TIME
- ACCOUNTING DATA IS MONITORED FOR EACH GROUP

THE USER

- DEFINED BY THE ACCOUNT MANAGER
  - NAME AND (PASSWORD)
  - (HOME GROUP NAME)
  - CAPABILITY LIST
  - MAXIMUM PRIORITY

- NOTE: ACCOUNTING DATA IS NOT KEPT BY USER, ONLY GROUPS AND ACCOUNTS ARE MONITORED

FILE ADDRESSING



LOG-ON GROUP

filename
or
filename/lockword

OUTSIDE LOG-ON GROUP

filename.groupname
or
filename/lockword.groupname

OUTSIDE LOG-ON ACCOUNT

filename.groupname.acctname
or
filename/lockword.groupname.acctname

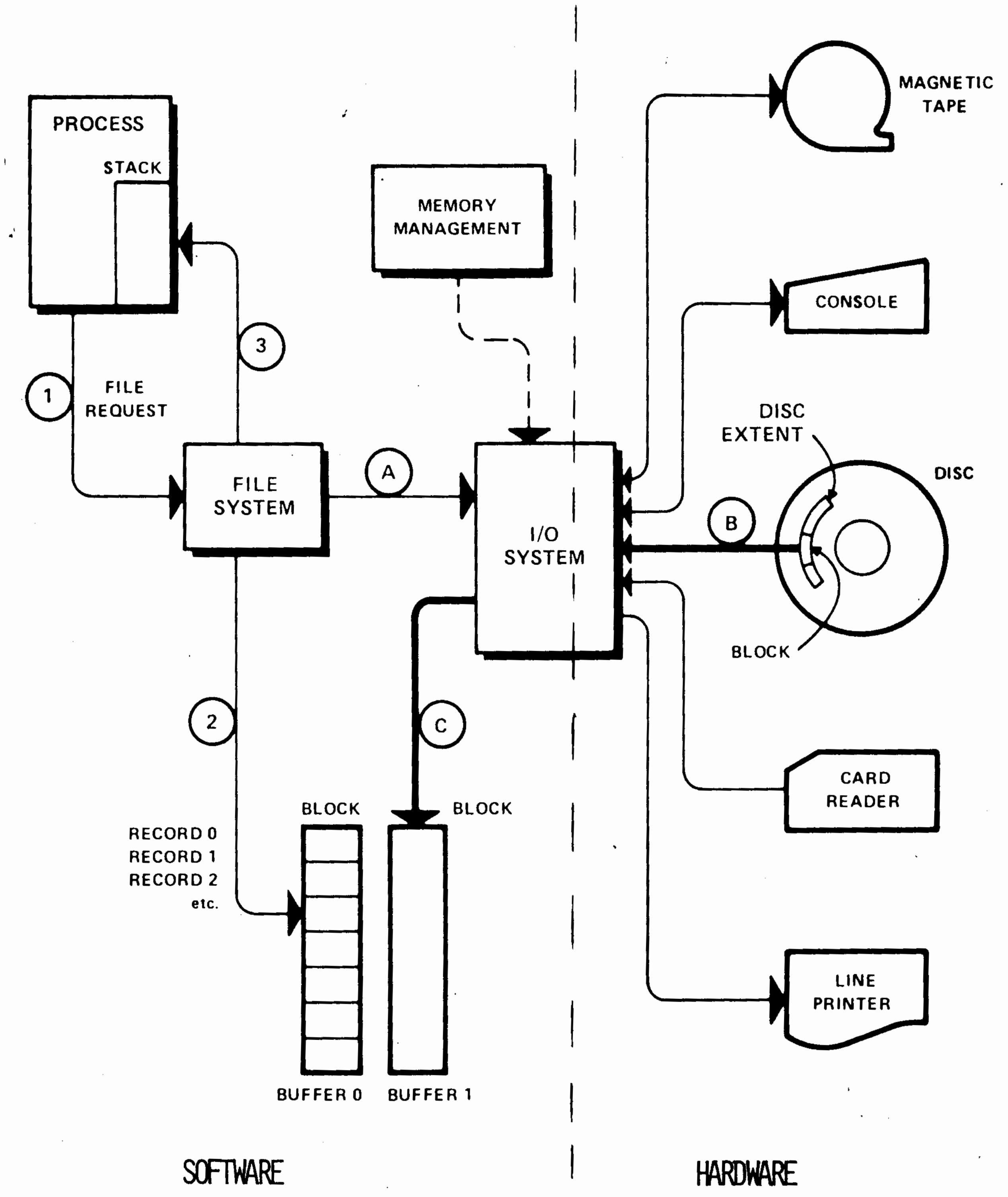
NO PASSWORDS

35 CHARACTERS MAXIMUM

FILE ADDRESSING EXAMPLES

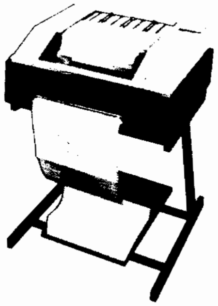
➧ EMPMAST	"EMPMAST" IS A FILE IN THE USERS LOG-ON GROUP.
➧ PAYROLL/KEEPOUT	"PAYROLL" IS A FILE IN THE USERS LOG-ON GROUP. THE FILE HAS A LOCKWORD "KEEPOUT".
➧ ACCTREC/LOCKY.FINANCE	"ACCTREC" IS A FILE IN THE "FINANCE" GROUP OF THE LOG-ON ACCOUNT. THE FILE HAS A LOCKWORD "LOCKY".
➧ JOBINFO.PUB.SYS	THE FILE "JOBINFO" IS A FILE IN THE "PUB" GROUP OF THE "SYS" ACCOUNT.

HP 3000 I/O SYSTEM

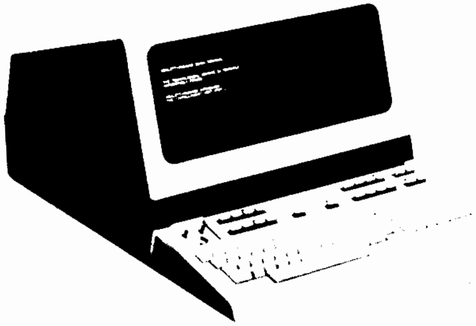


DEVICE

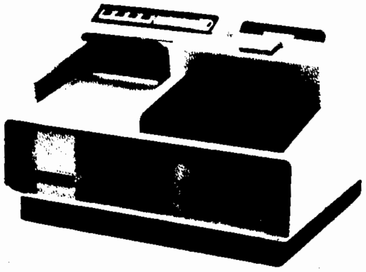
LINE PRINTER



TERMINALS



CARD READER



FILES

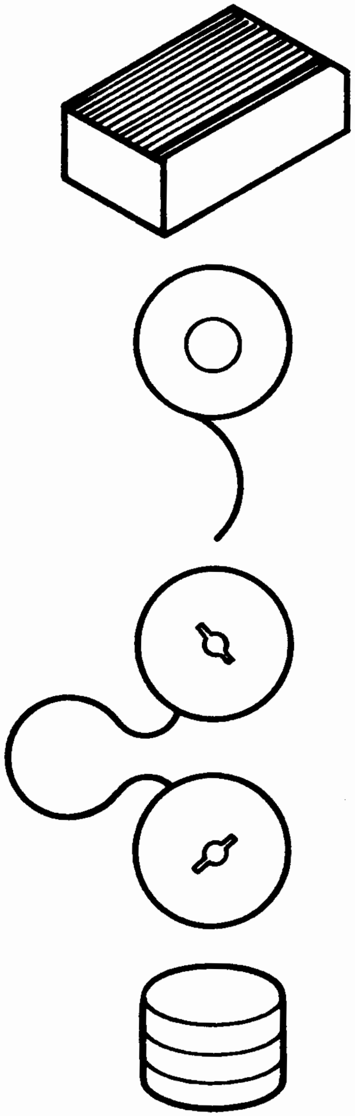
DATA

CARD DECKS

PAPER TAPE

MAGNETIC TAPE

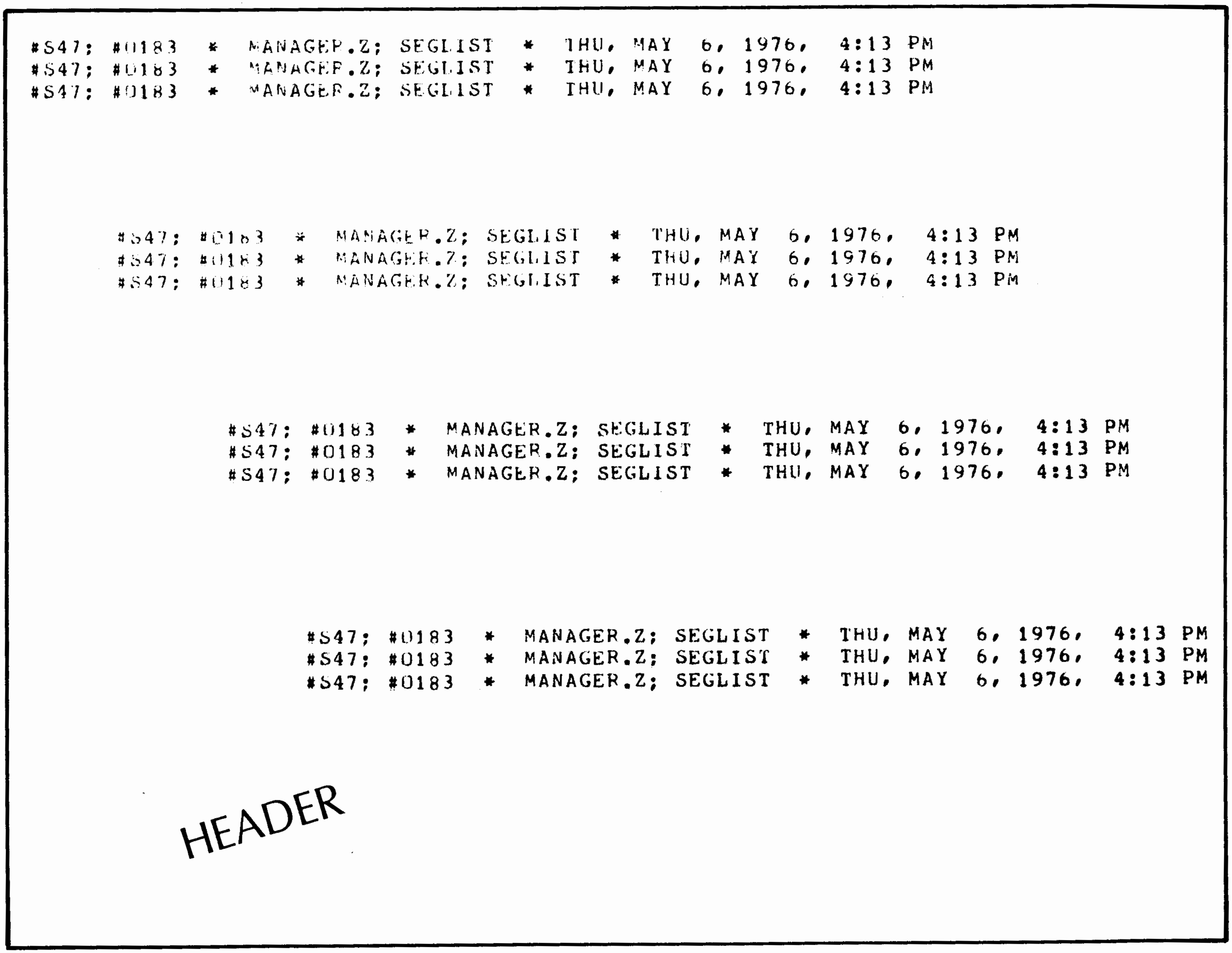
DISC



DEVICE

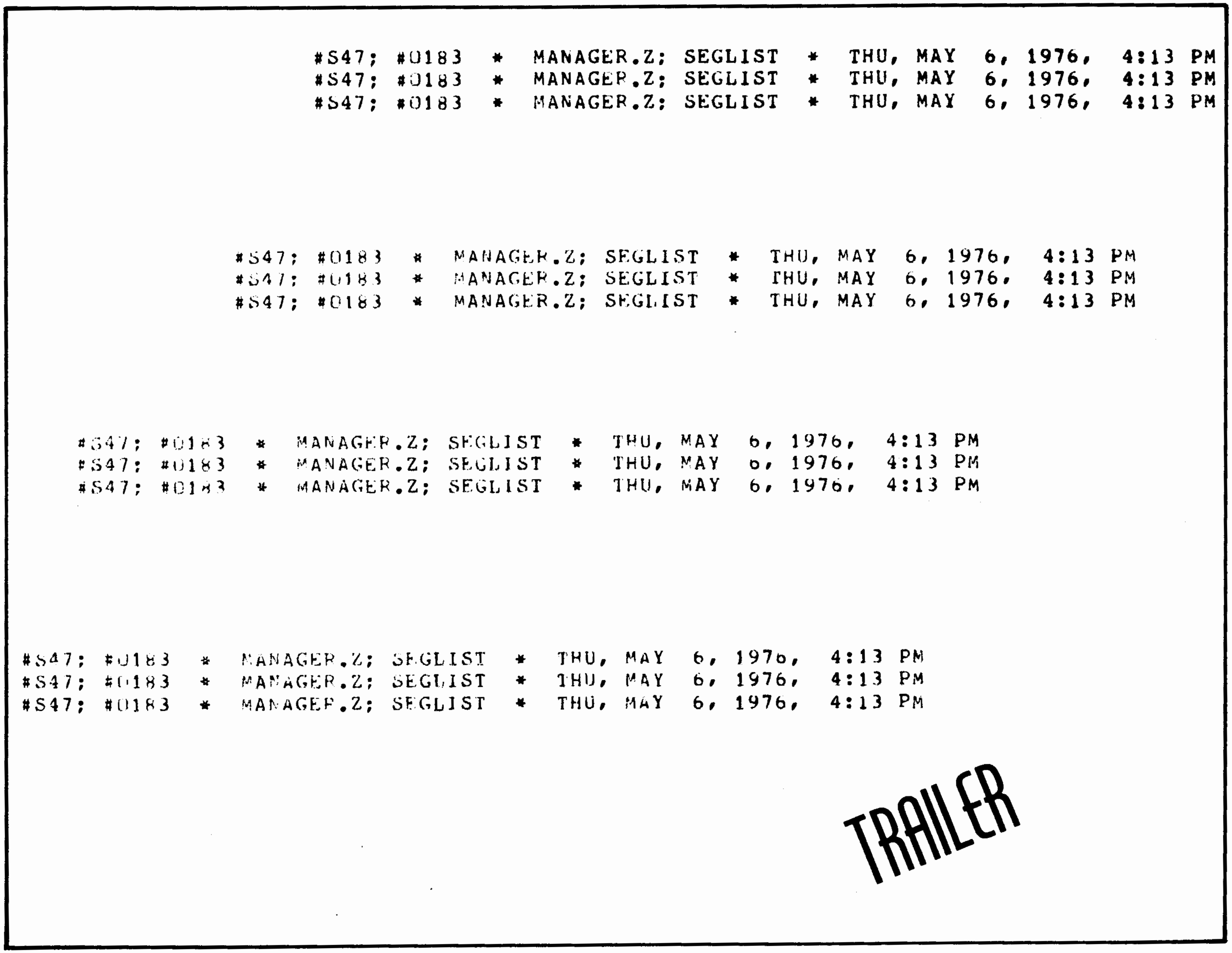
logical device number		device class name
1	(SYSTEM DISC)*	SYSDISC
2	(ADDITIONAL DISC)	DISC, SPOOL
3	(OPERATOR CONSOLE)*	CONSOLE
4		
5	(CARD READER)	CARD
6	(LINE PRINTER)	LP
7	(MAGNETIC TAPE)*	TAPE

* REQUIRED



```
#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




      #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
      #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
      #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




              #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
              #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
              #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




                    #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                    #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                    #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
```

HEADER



```
                                   #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                                   #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                                   #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




                             #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                             #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                             #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




                #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
                #S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM




#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
#S47; #0183  *  MANAGER.Z; SEGLIST  *  THU, MAY  6, 1976,  4:13 PM
```

TRAILER

MPE "JOB CONTROL" COMMAND ERRORS

ERR errnum [,detail] [message]

errnum	
Ø – 19	General
2Ø – 47	Command Syntax
48 – 99	Specific Commands
1ØØ – 199	File System
2ØØ – 249	CREATE/Loader
25Ø	Segmenter

EXAMPLE:

```
:FILE PRNT;DEV=LP,,2;REC=–80,ASCII
ERR 30,8
INVALID NUMBER
```

FILE INFORMATION DISPLAY 'TOMBSTONE'

```
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NUMBER #       IS UNDEFINED.              !
!  ERROR NUMBER: 56    RESIDUE: Ø                 !
!  BLOCK NUMBER: Ø              NUMREC: Ø         !
+-------------------------------------------------+
```

FOR FILES NOT YET OPENED

OR

"FOPEN" FAILURE

```
+-F-I-L-E---I-N-F-O-R-M-A-T-I-O-N---D-I-S-P-L-A-Y+
!  FILE NAME IS FTN05                             !
!  FOPTIONS: SYS,A,   $STDIN,U,N,FEQ              !
!  AOPTIONS:   INPUT,SREC,NOLOCK,DEF,NOBUFF       !
!  DEVICE TYPE: 16      DEVICE SUBTYPE: 0         !
!  LDEV: 11       DRT: 18        UNIT: 0          !
!  RECORD SIZE: 72      BLOCK SIZE: 72   (BYTES)  !
!  EXTENT SIZE: 0       MAX EXTENTS: 0            !
!  RECPTR: 0            RECLIMIT: 0               !
!  LOGCOUNT: 0             PHYSCOUNT: 0           !
!  EOF AT: 0            LABEL ADDR: %01300000000  !
!  FILE CODE: 0       ID IS           ULABELS: 0  !
!  PHYSICAL STATUS: 0000101100000000              !
!  ERROR NUMBER: 0      RESIDUE: 0                !
!  BLOCK NUMBER: 0              NUMREC: 1         !
+-------------------------------------------------+
```

END OF FILE

OR

IRRECOVERABLE FILE ERROR

BREAKABLE AND NON-BREAKABLE COMMANDS

NON-BREAKABLE		BREAKABLE	
		SUSPENDED	ABORTED
:ABORT	:ALTSEC	:BASIC	:LISTF
:BUILD	:BYE	:COBOL	:HELLO *
:COMMENT	:CONTINUE	:COBOLGO	:REPORT
:DATA	:EOD	:COBOLPREP	:STORE **
:EOJ	:FILE	:RPG	:RESTORE **
:FREERIN	:GETRIN	:RPGGO	:SHOWJOB
:JOB	:PTAPE	:RPGPREP	:SHOWDEV
:PURGE	:RELEASE	:RJE	:STREAM
:RENAME	:RESET	:BASICOMP	:SHOWIN
:RESUME	:SAVE	:BASICGO	:SHOWOUT
:SECURE	:SHOWTIME	:BASICPREP	
:SPEED	:TELL	:EDITOR	
:TELLOP	:SETDUMP	:FORTRAN	
:RESETDUMP	:SETMSG	:FORTGO	
		:FORTPREP	
		:PREP	
		:PREPRUN	
		:RUN	
		:SEGMENTER	
		:SPL	
		:SPLGO	
		:SPLPREP	
		:STAR	

* Breaking :HELLO will suppress the welcome message, if any exists.

**:STORE/:RESTORE, when broken, will stop after completing the current file and will suppress further output.

MPE INTRODUCTION REVIEW QUESTIONS

1. A session is initiated with a :____________ command; and terminated with a :______ command.

2. A job is initiated with a : ____________command and terminated with a :______command.

3. A session normally runs in the ____________ queue.

4. A job normally runs in the __________ queue.

5. The value of TERM=parameter in the Hello command should be ____________ when using a HP 2640 terminal,

6. a. Control__________ deletes the last character typed on the terminal.
b. Control__________ deletes the current line.

7. Check which of the following are valid commands:

		VALID	INVALID
A.	:JOB PAYRL, MGR.FINAPP		
B.	:HELLO S1.COMUTLZ;TERM=4;PRI=CQ		
C.	:HELLO MGR,COMUTLZ		
D.	:HELLO S1.COMUTLZ,G1		
E.	:HELLO		
F.	:DATA PAYRL,MGR.FINAPP;YTD		

8. A group of instructions is called a ______________ segment.

9. The users data area is called a_________________ or _________________

10. ______________ is not swapped.

11. ______________ is swappable.

ANSWERS TO MPE INTRODUCTION REVIEW QUESTIONS

1. HELLO, BYE

2. JOB, EOJ

3. CS

4. DS

5. 10

6. a. H
   b. X

7. A. VALID
   B. INVALID; CQ IS NOT A VALID PRIORITY
   C. INVALID; USER AND ACCOUNT MUST BE SEPARATED BY A PERIOD (.)
   D. VALID
   E. INVALID; username.acctname ARE REQUIRED.
   F. VALID

8. CODE

9. STACK OR DATA SEGMENT

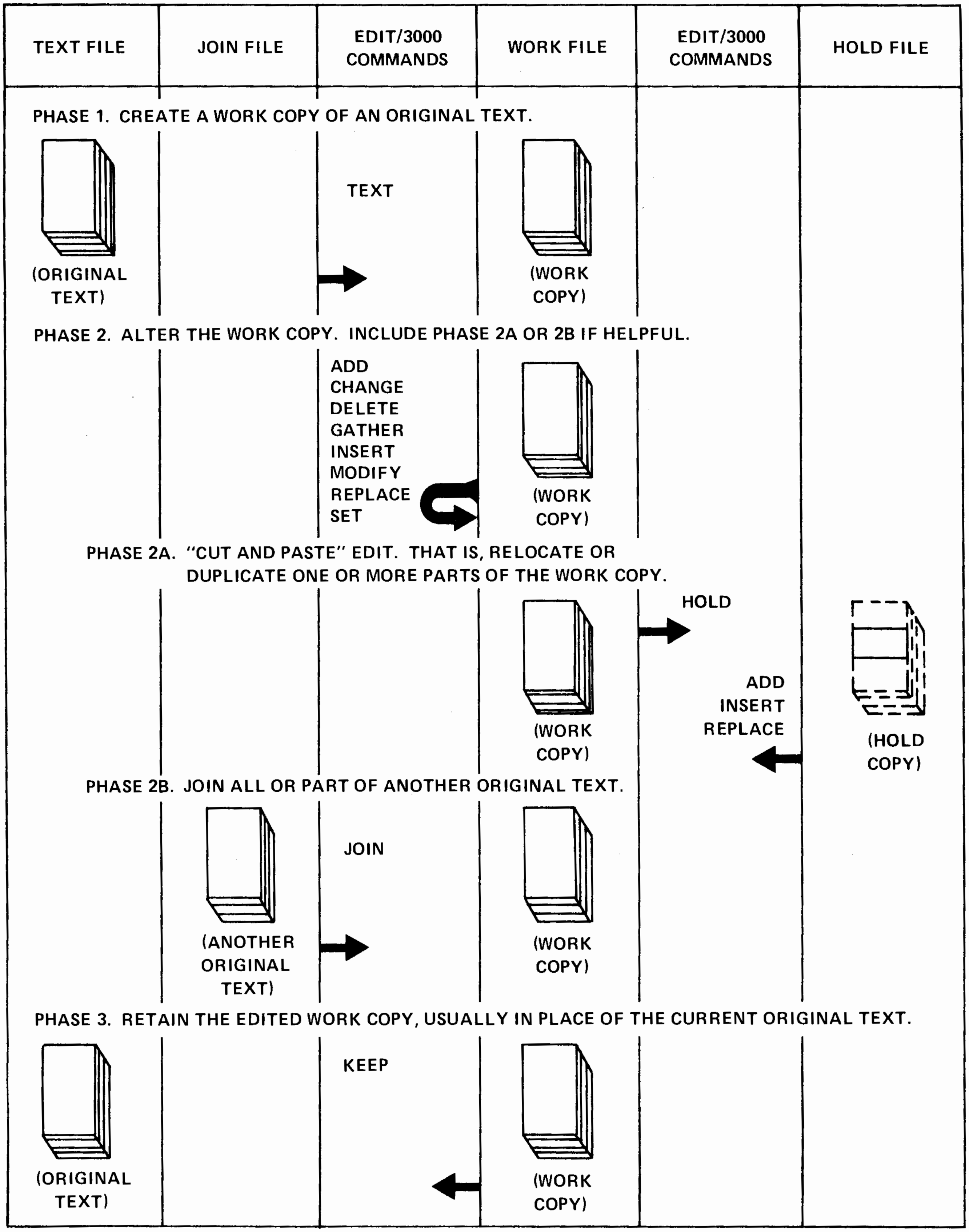
10. CODE

11. DATA

LAB #2 (LANGUAGES)

1. USING THE USER AND ACCOUNT NAME OBTAINED EARLIER FROM THE INSTRUCTOR, LOG ON THE SYSTEM.

2. COMPLETE SECTION 2 (FORTRAN) OR SECTION 3 (COBOL) OR SECTION 4 (BASIC) OF USING THE HP 3000, A GUIDE FOR THE TERMINAL USER.



TEXT FILE

JOIN FILE

EDIT/3000 COMMANDS

WORK FILE

EDIT/3000 COMMANDS

HOLD FILE

PHASE 1. CREATE A WORK COPY OF AN ORIGINAL TEXT.

(ORIGINAL TEXT)

TEXT

(WORK COPY)

PHASE 2. ALTER THE WORK COPY. INCLUDE PHASE 2A OR 2B IF HELPFUL.

ADD
CHANGE
DELETE
GATHER
INSERT
MODIFY
REPLACE
SET

(WORK COPY)

PHASE 2A. "CUT AND PASTE" EDIT. THAT IS, RELOCATE OR DUPLICATE ONE OR MORE PARTS OF THE WORK COPY.

(WORK COPY)

HOLD

ADD
INSERT
REPLACE

(HOLD COPY)

PHASE 2B. JOIN ALL OR PART OF ANOTHER ORIGINAL TEXT.

(ANOTHER ORIGINAL TEXT)

JOIN

(WORK COPY)

PHASE 3. RETAIN THE EDITED WORK COPY, USUALLY IN PLACE OF THE CURRENT ORIGINAL TEXT.

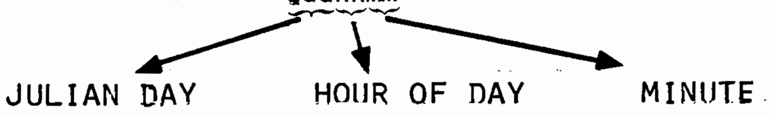
(ORIGINAL TEXT)

KEEP

(WORK COPY)

WORK FILE

- CREATED BY ADD, TEXT AND JOIN COMMANDS
- EDIT/3000 OPERATES ONLY ON WORK FILE
- NAME OF WORK FILE IS Kdddhhmm



JULIAN DAY HOUR OF DAY MINUTE

- RECOVERABLE AFTER SYSTEM CRASH (CHECK POINTERS)
  - LISTF TO LOCATE KFILENAME
  - /T KFILENAME
  - /K NEWFILENAME
  - /T NEWFILENAME

TEXT EDITOR

TEXT INPUT COMMANDS

- TEXT
- JOIN
- ADD
- INSERT

TEXT COMMAND

TO COPY ALL OR PART OF AN MPE/3000 FILE IN THE WORK FILE

```
T[EXT] filename         ⎡(linenumber/linenumber)⎤
                        ⎣(#recnum/#recnum)      ⎦

                         [,UNN [UMBERED] ]
```

TEXT COMMAND

EXAMPLE

TEXT INPUT FROM EXTERNAL DEVICE

```
T[EXT] filename    [(linenumber/linenumber)]
                   [(#recnum/#recnum)      ]

                    [,UNN [UMBERED] ]
```

```
:FILE CDIN;DEV=CARD                               FILE COMMAND

:EDITOR

HP 32201A.4.03 EDIT/3000  FRI, MAY 7, 1976, 11:23 AM


/TEXT *CDIN,UNN      <-------------        BACK-REFERENCE-UNNUMBERED
/LIST ALL
  1     THIS IS LINE ONE
  2     THIS IS LINE TWO
  3     THIS IS LINE THREE
  4     THIS IS LINE FOUR
```

TEXT COMMAND

```
T[EXT] filename   ⎡(linenumber/linenumber)⎤   [,UNN [UMBERED] ]
                  ⎣(#recnum/#recnum)      ⎦
```

```
 / TEXT COBLSORT
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
```

TEXT WILL CLEAR WORK FILE

example using disk file input

```
 / LIST ALL
      1            $CONTROL USLINIT
      2             IDENTIFICATION DIVISION.
      3             PROGRAM-ID. SORT-PROCEDURE.
      4             ENVIRONMENT DIVISION.
      5             DATA DIVISION.
      6             PROCEDURE DIVISION.
      7             BEGIN.
      8             EXIT PROGRAM.
/
```

join command

ADD ALL OR PART OF A USER DISC FILE TO THE WORK FILE.

```
J [OIN]  [Q]  filename          ┌(linenumber   [/linenumber] )┐
                                └(#recnum  [/#recnum] )       ┘

                                     [TO linenumber]
                                                         [,UNN]
                                     [BY increment]
```

JOIN COMMAND

```
J [OIN] [Q] filename     [(linenumber [/linenumber] )]
                         [(#recnum [/#recnum] )      ]

                              [TO linenumber]
                                                  [,UNN]
                              [BY increment]
```

JOIN WILL LIST

```
/JOIN COBLSORT
   1      $CONTROL USLINIT
   2       IDENTIFICATION DIVISION.
   3       PROGRAM-ID.  SORT-PROCEDURE.
   4       ENVIRONMENT DIVISION.
   5       DATA DIVISION.
   6       PROCEDURE DIVISION.
   7       BEGIN.
   8       EXIT PROGRAM.
```

JOIN COMMAND

```
J[OIN] [Q] filename      ⎡(linenumber  [/linenumber] )⎤
                         ⎣(#recnum [/#recnum] )       ⎦
                               [TO linenumber]
                                                 [,UNN]
                               [BY increment]
```

JOIN WILL NOT CLEAR WORK FILE

```
/JOIN SHORTF
   9       THIS IS LINE ONE
  1Ø       THIS IS LINE TWO
  11       THIS IS LINE THREE
  12       THIS IS LINE FOUR
```

```
/LIST ALL
    1             $CONTROL USLINIT
    2               IDENTIFICATION DIVISION.
    3               PROGRAM-ID.  SORT-PROCEDURE.
    4               ENVIRONMENT DIVISION.
    5               DATA DIVISION.
    6               PROCEDURE DIVISION.
    7               BEGIN.
    8               EXIT PROGRAM.
    9   THIS IS LINE ONE
   1Ø   THIS IS LINE TWO
   11   THIS IS LINE THREE
   12   THIS IS LINE FOUR
```

ADD COMMAND

TO ENTER TEXT INTO THE WORK FILE FROM
THE STANDARD INPUT DEVICE OR FROM THE HOLD FILE.

```
A [DD]  [Q]          [linenumber] [,HOLD [Q]  [,NOW]
```

ADD COMMAND

A[DD] [Q] [line number] [,HOLD[Q]] [,NOW]

```
/ADD
     1    THIS IS AN EXAMPLE
     2    OF DATA INPUT THROUGH
     3    THE ADD COMMAND OF
     4    THE HP 3ØØØ TEXT EDITOR
     5    //WILL TERMINATE THE
     6    ADD COMMAND ONLY IF
     7    IT ENDS THE LINE
     8    //
...
```

IF EMPTY WORK FILE ADD STARTS AT LINE ONE

//TERMINATES ADD

ADD WILL APPEND TO END OF WORK FILE

```
/ADD
     8    ADD WILL DEFAULT THE
     9    LINE NUMBER TO THE END OF THE TEXT BUFFER
    1Ø    //
...
```

ADD COMMAND

A[DD] [Q] [line number] [,HOLD[Q]] [,NOW]

CAN IMPLY A LINE INCREMENT

```
/ADD 5.Ø1
      5.Ø1  THIS IS HOW
      5.Ø2  YOU CAN SPECIFY
      5.Ø3  A LINE INCREMENT
      5.Ø4  LESS THAN 1.Ø
      5.Ø5  IT
      5.Ø6  WILL
      5.Ø7  AUTOMATICALLY
      5.Ø8  TAKE
      5.Ø9  THE
      5.1   NEW
      5.11  INCREMENT
      5.12  SPECIFIED IN THE COMMAND ITSELF.
      5.13  //
. . .
```

ADD WILL NOT REPLACE LINES

```
/ADD 5.Ø1
*15*
COMMAND WILL NOT REPLACE OR INTERLEAVE LINES
```

INSERT COMMAND

INSERT TEXT INTO THE WORK FILE FROM THE STANDARD INPUT DEVICE OR FROM THE HOLD FILE.

```
I [INSERT] [Q] [position] [BY increment] [,HOLD[Q][,NOW]
```

```
/T SHORTF
/LIST ALL
    1    THIS IS LINE ONE
    2    THIS IS LINE TWO
    3    THIS IS LINE THREE
    4    THIS IS LINE FOUR
```

INSERT COMMAND

```
I[NSERT] [Q] [position] [BY  increment  ] [,HOLD[Q] [,NOW]
```

CAN INSERT WHOLE LINES

```
/INSERT 2
    2       THIS IS LINE TWO
            ↑NOW THIS IS THE NEW LINE TWO
    2.1     AND THIS WILL BE 2.1
    2.2     //
THIS IS LINE TWO
```

PRE-EXISTING LINES WILL SLIP DOWN IN SEQUENCE

```
/LIST ALL
    1       THIS IS LINE ONE
    2       NOW THIS IS THE NEW LINE TWO
    2.1     AND THIS WILL BE 2.1
    2.2     THIS IS LINE TWO
    3       THIS IS LINE THREE
    4       THIS IS LINE FOUR
```

CAN INPUT PART OF A LINE //WILL KEEP REST OF LINE

```
/INSERT 4(9)
    4       THIS IS LINE FOUR
                  ↑THE LINE AFTER THREE.  IT IS //
/LIST 4
    4       THIS IS THE LINE AFTER THREE.  IT IS LINE FOUR
```

INSERT COMMAND

```
/TEXT SHORTF
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES
/LIST ALL

     1      THIS IS LINE ONE
     2      THIS IS LINE TWO
     3      THIS IS LINE THREE
     4      THIS IS LINE FOUR

/INSERT 4(+4)
     4      THIS IS LINE FOUR
                 ↑NOW //

IS LINE FOUR
/LIST 4
     4      THIS NOW IS LINE FOUR

/INSERT 4(6)
     4      THIS NOW IS LINE FOUR
                 ↑
     5      //
NOW IS LINE FOUR
/LIST ALL
     1      THIS IS LINE ONE
     2      THIS IS LINE TWO
     3      THIS IS LINE THREE
     4      THIS
     5      NOW IS LINE FOUR
```

INSERT AT 4TH NON BLANK CHARACTER

INSERT AT 6TH COLUMN POSITION

CARRIAGE RETURN WILL SPLIT LINE

RANGE EXPRESSIONS

```
/LIST ALL
      1          THIS IS LINE ONE                ALL
      2          THIS IS LINE TWO
      3          THIS IS LINE THREE
      4          THIS IS LINE FOUR


/LIST FIRST
      1          THIS IS LINE ONE                FIRST

/LIST LAST
      4          THIS IS LINE FOUR               LAST
```

RANGE EXPRESSIONS

*****

```
/LIST *
     4          THIS IS LINE FOUR
```

LAST-1

```
/LIST LAST-1
     3          THIS IS LINE THREE
```

FIRST+2

```
/LIST FIRST+2
     3          THIS IS LINE THREE
```

RANGE EXPRESSIONS

*/"THIRD"

```
/ADD
     1          THIS IS THE FIRST LINE
     2          THIS IS THE SECOND LINE
     3          THIS IS THE THIRD LINE
     4          THIS IS THE LAST LINE
     5          ...


/FIND FIRST
     1          THIS IS THE FIRST LINE
                ↑ (1    )
/DELETE */"THIRD"
     1          THIS IS THE FIRST LINE
     2          THIS IS THE SECOND LINE
     3          THIS IS THE THIRD LINE
/LIST ALL
     3            LINE
     4          THIS IS THE LAST LINE
/FIND FIRST
     3          ↑ LINE
```

RANGE EXPRESSIONS

/ +1

```
/DELETE  */*+1
     3          LINE
     4        THIS IS THE LAST LINE

***  WARNING *** WORK FILE IS EMPTY.

/LIST ALL

/TEXT SHORTF
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?  YES

/LIST ALL
     1        THIS IS LINE ONE
     2        THIS IS LINE TWO
     3        THIS IS LINE THREE
     4        THIS IS LINE FOUR

/FIND 3
     3        THIS IS LINE THREE
              ↑(1   )
```

RANGE EXPRESSIONS

```
/DELETE  */*(5)
    3          THIS IS LINE THREE
```

/ (5)

ABSOLUTE COLUMN

```
/LIST 3
    3          IS LINE THREE


/FIND 4
    4          THIS IS LINE FOUR
               ↑ (1     )


/DELETE  */*(+5)
    4          THIS IS LINE FOUR
```

/ (+5)

NON-BLANK CHARACTERS

```
/LIST 4
    4           LINE FOUR
```

TEXT EDITOR

TEXT MANIPULATION COMMANDS

- CHANGE
- DELETE
- REPLACE
- GATHER
- MODIFY

CHANGE COMMAND

CHANGE EXISTING CONTENTS OF THE WORK FILE

$$\text{C[HANGE] [Q]} \left\{ \begin{array}{ll} \text{string} & \\ \text{col} & \text{[/col]} \end{array} \right\} \text{TO string [IN rangelist]}$$

CHANGE COMMAND

$$\text{C[HANGE] [Q]} \left\{ \begin{array}{ll} \text{string} & \\ \text{col} & \text{[/col]} \end{array} \right\} \text{TO string [IN rangelist]}$$

```
/TEXT SHORTF
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?  YES
/LIST ALL

     1        THIS IS LINE ONE
     2        THIS IS LINE TWO
     3        THIS IS LINE THREE
     4        THIS IS LINE FOUR
```

```
/CHANGE "ONE" TO "1" IN 1
     1        THIS IS LINE 1
```

CHANGE A STRING IN A LINE

CHANGE COMMAND

$$\text{C[HANGE] [Q]} \begin{Bmatrix} \text{string} \\ \text{col[/col]} \end{Bmatrix} \text{TO string [IN rangelist]}$$

```
/CHANGE 1/4 TO "HERE" IN 2                      CHANGE A COLUMN
     2       HERE IS LINE TWO

/CHANGE 1 TO "NOW   " IN 3                      PREFIX A LINE
     3       NOW THIS IS LINE THREE

/CHANGE "LINE" TO "ITEM" IN ALL                 CHANGE ALL STRINGS
     1       THIS IS ITEM 1                     IN A RANGE
     2       HERE IS ITEM TWO
     3       NOW THIS IS ITEM THREE
     4       THIS IS ITEM FOUR
```

DELETE COMMAND

TO DELETE CHARACTER AND/OR LINES FROM THE WORK FILE.

D[ELETE] [Q] [rangelist]

DELETE COMMAND

D[ELETE] [Q] [range list]

```
/TEXT COBLSORT
/LIST ALL

     1             $CONTROL USLINIT
     2              IDENTIFICATION DIVISION.
     3              PROGRAM-ID.  SORT-PROCEDURE.
     4              ENVIRONMENT DIVISION.
     5              DATA DIVISION.
     6              PROCEDURE DIVISION.
     7              BEGIN.
     8              EXIT PROGRAM.
```

DELETE COMMAND

D[ELETE] [Q] [range list]

DELETE COLUMNS IN MULTIPLE LINES

```
/DELETE 2(15)/6(2Ø)
    2                  IDENTIFICATION DIVISION.
    5                  DATA DIVISION.
    6                  PROCEDURE DIVISION.
/LIST ALL
    1                 $CONTROL USLINIT
    2                  IDENTI
    6       VISION.
    7                  BEGIN.
    8                  EXIT PROGRAM.
```

DELETE RANGE LIST

```
/DELETE 2/6,8
    2                  IDENTI
    6       VISION.
    8                  EXIT PROGRAM.
/LIST ALL
    1                 $CONTROL USLINIT
    7                  BEGIN.
```

REPLACE COMMAND

REPLACE ONE OR MORE LINES IN THE WORK FILE FROM
THE STANDARD INPUT DEVICE OR FROM THE HOLD FILE.

```
R[EPLACE] [Q] [rangelist] [,HOLD [Q] [,NOW] ]
```

REPLACE COMMAND

R[EPLACE] [Q] [range list] [,HOLD[Q[[,NOW]]

REPLACE COMPLETE LINE

```
/TEXT SHORTF
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?  YES
/LIST ALL
     1         THIS IS LINE ONE
     2         THIS IS LINE TWO
     3         THIS IS LINE THREE
     4         THIS IS LINE FOUR
/REPLACE 3
     3         THIS IS LINE THREE
     3         NOW THIS IS A WHOLE NEW LINE NUMBER THREE
/LIST ALL
     1         THIS IS LINE ONE
     2         THIS IS LINE TWO
     3         NOW THIS IS A WHOLE NEW LINE NUMBER THREE
     4         THIS IS LINE FOUR
```

REPLACE COMMAND

R[EPLACE] [Q] [range list] [,HOLD [Q] [,NOW]]

REPLACE A RANGE OF LINES

```
/REPLACE 1,3/4
 1      THIS IS LINE ONE
 1      ***LINE ONE***
 3      NOW THIS IS A WHOLE NEW LINE NUMBER THREE
 3      ***LINE THREE***
 4      THIS IS LINE FOUR
 4      ***LINE FOUR***
/L ALL
 1      ***LINE ONE***
 2      THIS IS LINE TWO
 3      ***LINE THREE***
 4      ***LINE FOUR***
```

GATHER COMMAND

MOVE PORTIONS OF TEXT FROM ONE LOCATION TO ANOTHER IN THE WORK FILE AND RENUMBER THE LINES. ALSO CAN BE USED TO RENUMBER ALL LINES IN THE WORK FILE.

```
                 ( range   { TO }   linenumber         )
                 (         { ,  }                      )
G[ATHER]  [Q]   {                                       }[  BY  [increment] ]
                 (         { TO }                      )
                 ( ALL [   { ,  }   [linenumber] ]     )
```

GATHER COMMAND

$$\text{G[ATHER] [Q]} \left\{ \begin{array}{l} \text{range} \left\{ \begin{array}{c} \text{TO} \\ , \end{array} \right\} \text{linenumber} \\ \\ \text{ALL [} \left\{ \begin{array}{c} \text{TO} \\ , \end{array} \right\} \text{[linenumber]]} \end{array} \right\} \text{[BY [increment]]}$$

```
/JOIN COBLSORT
    1                   $CONTROL USLINIT
    2                    IDENTIFICATION DIVISION.
    3                    PROGRAM-ID.  SORT-PROCEDURE.
    4                    ENVIRONMENT DIVISION.
    5                    DATA DIVISION.
    6                    PROCEDURE DIVISION.
    7                    BEGIN.
    8                    EXIT PROGRAM.
```

MOVE RANGE TO NEW RANGE WITH NEW INCREMENT

```
/GATHER 3/4 TO 5.ØØ1
    3     =>    5.ØØ1
    4     =>    5.ØØ2
/LIST ALL
    1                   $CONTROL USLINIT
    2                    IDENTIFICATION DIVISION.
    5                    DATA DIVISION.
    5.ØØ1                PROGRAM-ID.  SORT-PROCEDURE.
    5.ØØ2                ENVIRONMENT DIVISION.
    6                    PROCEDURE DIVISION.
    7                    BEGIN.
    8                    EXIT PROGRAM.
```

GATHER COMMAND

```
                       ( range    { TO }    linenumber     )
                       (          { ,  }                   )
G[ATHER] [Q]           (                                   ) [  BY   [increment] ]
                       (          { TO }                   )
                       ( ALL  [   { ,  }    [linenumber] ] )   ,
```

GATHER ALL WILL RENUMBER TEXT

```
/GATHER ALL
/LIST ALL
    1              $CONTROL USLINIT
    2               IDENTIFICATION DIVISION.
    3               DATA DIVISION.
    4               PROGRAM-ID.  SORT-PROCEDURE.
    5               ENVIRONMENT DIVISION.
    6               PROCEDURE DIVISION.
    7               BEGIN.
    8               EXIT PROGRAM.
/
```

MODIFY COMMAND

MODIFY TEXT IN THE WORK FILE USING ONE OR MORE SUBCOMMANDS. (DELETE, INSERT, AND REPLACE) OF THIS COMMAND.

```
M[ODIFY] [Q] [rangelist]
```

MODIFY COMMAND

M[ODIFY] [Q] [rangelist]

```
/LIST ALL
    1          $CONTROL USLINIT
    2           IDENTIFICATION DIVISION.
    3           DATA DIVISION.
    4           PROGRAM-ID.  SORT-PROCEDURE.
    5           ENVIRONMENT DIVISION.
    6           PROCEDURE DIVISION.
    7           BEGIN.
    8           EXIT PROGRAM.
/MODIFY 2
MODIFY     2
        IDENTIFICATION DIVISION.
               DDD
        IDENTIFTION DIVISION.
          D              D
        IDION.
```

"D" WILL DELETE CHARACTER

MULTIPLE "D" WILL BRACKET DELETE RANGE

```
/MODIFY 4
MODIFY     4
        PROGRAM-ID.  SORT-PROCEDURE.
                                   I-CLASS-DEMO
        PROGRAM-ID.  SORT-PROCEDURE-CLASS-DEMO.
```

"I" WILL INSERT STRING

MODIFY COMMAND

M[ODIFY] [Q] [rangelist]

```
/MODIFY 7
MODIFY      7
          BEGIN.
          RSTART-HERE.
          START-HERE.


/MODIFY 8
MODIFY      8
          EXIT PROGRAM.
              R-HERE.XX
          EXIT-HERE.XX.
//
MODIFY      8
          EXIT PROGRAM.
```

"R" WILL REPLACE CHARACTERS

//WILL RESTORE ORIGINAL LINE CONTENTS

MODIFY COMMAND

M[ODIFY] [Q] [rangelist]

MODIFY A RANGE OF LINES

```
/MODIFY 2/4
MODIFY         2
          IDION.
          RIDENTIFICATION DIVISION.
          IDENTIFICATION DIVISION.

MODIFY         3
          DATA DIVISION.
                                I THIS IS A COMMENT
          DATA DIVISION.         THIS IS A COMMENT

MODIFY         4
          PROGRAM-ID.  SORT-PROCEDURE-CLASS-DEMO.
                                             DDDDDDDDDDDI.
          PROGRAM-ID.  SORT-PROCEDURE. .
                                              R
          PROGRAM-ID.  SORT-PROCEDURE.



/LIST ALL
    1                 $CONTROL USLINIT
    2                  IDENTIFICATION DIVISION.
    3                  DATA DIVISION.          THIS IS A COMMENT
    4                  PROGRAM-ID.  SORT-PROCEDURE.
    5                  ENVIRONMENT DIVISION.
    6                  PROCEDURE DIVISION.
    7                  START-HERE.
    8                  EXIT PROGRAM.
```

TEXT EDITOR
TEXT OUTPUT COMMANDS

/KEEP

/LIST

KEEP COMMAND

SAVE ALL OR PART OF THE WORK FILE INTO A USER FILE.

```
FORM I     K[EEP]     filename      [(range)]     [,UNN[UMBERED] ]

FORM II    K[EEP]     Q filename
```

KEEP COMMAND

FORM I K[EEP] filename [(range)] [,UNN[UMBERED]]

FORM II K[EEP] Q filename

CREATES A NEW FILE WITH THE
CONTENTS OF THE WORK FILE

```
/KEEP DUMMYFL
```

DUMMYFL ALREADY EXISTS – RESPOND YES TO PURGE OLD AND THEN KEEP

```
PURGE  OLD?Y
```

WILL WARN IF FILE ALREADY EXISTS

LIST COMMAND

LIST ALL OR PART OF THE WORK FILE TO THE STANDARD LIST FILE OR A PREVIOUSLY SPECIFIED FILE.

```
L[IST] [Q] [range] [,UNN[UMBERED] [,OFFLINE] [,TRANSLATE] [,NOTEXT]
```

```
:FILE LIST;DEV=LP
:EDITOR *LIST
/TEXT SHORTF
/LIST ALL,OFFLINE
```

```
:EDITOR
/TEXT SHORTF
 (break)
:FILE EDTLIST;DEV=LP
 (resume)
/LIST ALL,OFFLINE
```

NOTE: EDTLIST IS formaldesignator FOR EDITOR OFFLINE LIST DEVICE

LIST COMMAND

```
L[IST] [Q]   [range]   [,UNN[UMBERED] ]  [,OFFLINE]  [,TRANSLATE]  [,NOTEXT]
```

```
/LIST ALL
     1         THIS IS LINE ONE
     2         THIS IS LINE TWO
     3         THIS IS LINE THREE
     4         THIS IS LINE FOUR


/LIST ALL OFFLINE
*OFFLINE LIST DEVICE NOT AVAILABLE
/LIST ALL OFFLINE
```

LIST A RANGE ON PRINTER

TEXT EDITOR
MISCELLANEOUS COMMANDS

- HOLD
- FIND
- SET
- VERIFY
- WHILE
- END

HOLD COMMAND/FILE

COPY PART OR ALL OF THE WORK FILE INTO THE HOLD FILE FOR SUBSEQUENT COPYING INTO ONE OR MORE LOCATIONS OF THE WORK FILE.

```
H[OLD] [Q] [range] [,APPEND]
```

HOLD COMMAND/FILE

H[OLD] [Q] [range] [,APPEND]

```
/ADD
    1      <<<<<<<<HOLD FILE CONTENTS -- LINE ONE>>>>>>>>
    2      <<<<<<<<HOLD FILE CONTENTS -- LINE TWO>>>>>>>>
    3      //
```

COPIES RANGE TO HOLD FILE

```
. . .
/HOLD 1/2
<<<<<<<<HOLD FILE CONTENTS -- LINE ONE>>>>>>>>
<<<<<<<<HOLD FILE CONTENTS -- LINE TWO>>>>>>>>
/TEXT SHORTF
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES
/ADD ,HOLD
    5      THIS LINE IS INPUT FROM THE TERMINAL
    6      //
```

CAN ADD HOLD FILE

```
. . .
    6      <<<<<<<<HOLD FILE CONTENTS -- LINE ONE>>>>>>>>
    7      <<<<<<<<HOLD FILE CONTENTS -- LINE TWO>>>>>>>>
. . .
/LIST ALL
    1      THIS IS LINE ONE
    2      THIS IS LINE TWO
    3      THIS IS LINE THREE
    4      THIS IS LINE FOUR
    5      THIS IS LINE IS INPUT FROM THE TERMINAL
    6      <<<<<<<<HOLD FILE CONTENTS -- LINE ONE>>>>>>>>
    7      <<<<<<<<HOLD FILE CONTENTS -- LINE TWO>>>>>>>>
```

CONTENTS AFTER OTHER LINE INPUT

IMMEDIATELY ADDS CONTENTS OF HOLD FILE

```
/ADD ,HOLD,NOW
    8      <<<<<<<<HOLD FILE CONTENTS -- LINE ONE>>>>>>>>
    9      <<<<<<<<HOLD FILE CONTENTS -- LINE TWO>>>>>>>>
. . .
```

HOLD COMMAND/FILE

H[OLD] [Q] [range] [,APPEND]

```
/LIST ALL
   1       THIS IS LINE ONE
   2       THIS IS LINE TWO
   3       THIS IS LINE THREE
   4       THIS IS LINE FOUR
   5       THIS LINE IS INPUT FROM THE TERMINAL
   6       <<<<<HOLD FILE CONTENTS - -LINE ONE>>>>>
   7       <<<<<HOLD FILE CONTENTS  - -LINE TWO>>>>>
   8       <<<<<HOLD FILE CONTENTS - -LINE ONE>>>>>
   9       <<<<<HOLD FILE CONTENTS  - -LINE TWO>>>>>
   /DELETE ALL
   IF IT IS OK TO CLEAR RESPOND "YES"
   CLEAR? Y
   /HOLD
```

HOLD WILL CLEAR HOLD FILE

FIND COMMAND

FIND A SPECIFIC POSITION OR A CHARACTER STRING IN
THE WORK FILE.

F[IND] [Q] $\begin{bmatrix}\text{range}\\\text{string}\end{bmatrix}$

FIND COMMAND

```
                                                  F[IND] [Q] [range ]
/TEXT SHORTF                                                 [string]
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR?  YES
/LIST ALL
    1        THIS IS LINE ONE
    2        THIS IS LINE TWO
    3        THIS IS LINE THREE     FIND A LINE
    4        THIS IS LINE FOUR
/FIND 4
    4        THIS IS LINE FOUR      FIND CURRENT LINE
             ↑(1   )
/FIND *
    4        THIS IS LINE FOUR      FIND FIRST LINE
             ↑(1   )
/FIND FIRST
    1        THIS IS LINE ONE
             ↑(1   )
/FIND "TWO"
    2        THIS IS LINE TWO       FIND A STRING
                        (14 )↑
/FIND "ONE"                         FIND COMMAND WILL
*21*                                SEARCH FROM
STRING NOT FOUND BEFORE LIMIT       POINTER TO END OF
/FIND FIRST                         WORK FILE
    1        THIS IS LINE ONE
             ↑(1   )
/FIND "ONE"/*(+12)                  SEARCH TO 12TH
    1        THIS IS LINE ONE       NON-BLANK
                        (14 )↑      CHARACTERS
```

FIND COMMAND

F[IND] [Q] [range
string]

```
/FIND FIRST
    1      THIS IS LINE ONE
           ↑(1   )
/FIND "ONE"/*(12)
*21*
STRING NOT FOUND BEFORE LIMIT
```

SEARCH TO 12TH CHARACTER

```
/FIND FIRST
    1      THIS IS LINE ONE
           ↑(1   )
/FIND "THREE"/*+3
    3      THIS IS LINE THREE
                   (14  )↑
```

WILL SEARCH A RANGE OF LINES

SET COMMAND

```
S[ET] [[,]FROM= line number ] [[,]DELTA= incr ] [[,]LEFT= col.num ]
 [[,]RIGHT= col.num ] [[,]LENGTH= col.num ] [[,] QUIETIDISPLAY ]
   [[,] SHORTILONG ] [[,] BATCHIPOLL ] [[,]DEPTH= limiter ]
   [[,]TIME[S]= limiter ] [[,]SIZE= integer ] [[,] FRONTIREAR ]
     [[,] FIXEDIVARIABLE ] [[,]FORMAT= COBOLIDEFAULT ]
```

CHANGE INCREMENT NUMBER

```
/SET DELTA=1Ø


/ADD
     1       THIS IS AN EXAMPLE
    11       OF A DIFFERENT
    21       LINE INCREMENT
    31       SET BY DELTA//
. . .
```

INHIBIT EDITOR VERBALIZATION

```
/SET QUIET
/DELETE 11/LAST
NUMBER OF LINES DELETED = 2

/SET DISPLAY

/DELETE 1
     1       THIS IS AN EXAMPLE
*** WARNING *** WORK FILE IS EMPTY.


/E
```

```
S[ET] [[,]FROM= line number ] [[,]DELTA= incr ] [[,]LEFT= col.num ]
 [[,]RIGHT= col.num ] [[,]LENGTH= col.num ] [[,] QUIET|DISPLAY ]
   [[,] SHORT|LONG ] [[,] BATCH|POLL ] [[,]DEPTH= limiter ]
   [[,]TIME[S]= limiter ] [[,]SIZE= integer ] [[,] FRONT|REAR ]
     [[,] FIXED|VARIABLE ] [[,]FORMAT= COBOL|DEFAULT ] .
```

```
/SET RIGHT=10                         RIGHT COLUMN POSITION SET


/SET SHORT

/A
    1     A B C D E F G H I J K L M N O P Q R S T
    2     1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0
    3     //
. . .
/L ALL
    1     A B C D E F G H I J                NOTE ONLY 10 COLUMNS
    2     1 2 3 4 5 6 7 8 9 0
/KEEP LINES
PURGE OLD?YES                         LONG REVERSES SET SHORT
```

SET COMMAND

VERIFY COMMAND

```
V[ERIFY] [  ALL [[,]FROM] [[,]DELTA] [[,]LEFT] [[,]RIGHT]
      [[,]LENGTH] [[,]QUIET] [[,]DISPLAY] [[,]SHORT]
        [[,]LONG] [[,]BATCH] [[,]POLL] [[,]DEPTH]
        [[,]TIME[S] [[,]SIZE] [[,]FRONT] [[,]REAR]
      [[,]FIXED] [[,]VARIABLE] [[,]FORMAT] [[,]FILES]
                       [[,]TOTAL]  ]
```

```
ADD
     1      THIS IS LINE ONE
     2      LINE TWO
     3      //
. . .
/VERIFY
     2      LINE TWO
            ↑(1   )
/VERIFY ALL
     2      LINE TWO
            ↑(1   )
```

VERIFY CURRENT POINTER

VERIFY COMMAND

VERIFY ALL VARIABLES THAT CAN BE SET

```
V[ERIFY] [  ALL [[,]FROM] [[,]DELTA] [[,]LEFT] [[,]RIGHT]
     [[,]LENGTH] [[,]QUIET] [[,]DISPLAY] [[,]SHORT]
        [[,]LONG] [[,]BATCH] [[,]POLL] [[,]DEPTH]
        [[,]TIME[S] [[,]SIZE] [[,]FRONT] [[,]REAR]
     [[,]FIXED] [[,]VARIABLE] [[,]FORMAT] [[,]FILES]
```

```
POLL = TRUE (I.E. BATCH = FALSE)
REAR = TRUE (I.E. FRONT = FALSE)
DELTA =       1
CURRENT DEPTH = Ø, THE DEPTH LIMIT = 1Ø
RIGHT = 1Ø
LENGTH = 72
LONG = TRUE (I.E. SHORT = FALSE)
TIME = 5Ø
TOTAL NUMBER OF CURRENT LINES = 2
FROM =        1
LEFT =  1
FIXED = TRUE (I.E. VARIABLE = FALSE)
SIZE = Ø
DISPLAY = TRUE (I.E. QUIET = FALSE)
FORMAT=DEFAULT
FILES:
    WORK:  KØ721ØØ6
    KEEP:
    TEXT:  SHORTF.GØØ.UTLZ      THU, MAR 13, 1975, 9:5Ø AM
    JOIN:  SHORTF.GØØ.UTLZ      THU, MAR 13, 1975, 9:44 AM
```

WHILE COMMAND

```
/TEXT EDLAB
/FIND FIRST
    1                                  WHAT IS A GOOD PROGRAM? *
          ↑(1   )
/
 WHILE
/    FINDQ "PROGRAMMER"
/    LIST *
   /FINDQ  "PROGRAMMER"
   /LIST *
    7      SUCCESSFUL, AND THIS COAL IS THE PRIMARY AIM OF A PROGRAMMER.
   /FINDQ  "PROGRAMMER"
   /LIST *
    9      BECOME MORE IMPORTANT AS A PROGRAMMER GAINS EXPERIENCE AND AS
THE
   /FINDQ  "PROGRAMMER"
   /LIST *
   19      PROGRAMMER MUST USE AS FEW INSTRUCTIONS AS POSSIBLE AND CHOOSE
   /FINDQ  "PROGRAMMER"
   /LIST *
   21      CYCLES THAT ARE REQUIRED.  ALSO THE PROGRAMMER MUST USE INPUT-
OUTPUT
   /FINDQ  "PROGRAMMER"
   /LIST *
   24      PROGRAMMER.  QUITE OFTEN A PROGRAMMER BECOMES SO ANXIOUS TO SA
VE
   /FINDQ  "PROGRAMMER"
   /LIST *
   32      DESIGNS A PROGRAM, THE PROGRAMMER SHOULD TRY TO ANTICIPATE AS
   /FINDQ  "PROGRAMMER"
   /LIST *
   52      THIS SITUATION HAS OCCURRED MANY TIMES.  IT IS UP TO THE PROGR
AMMER,
   /FINDQ  "PROGRAMMER"
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
 -/LIST *
/
```

WHILE INITIALIZES SPECIAL INTERPRETER

SECOND COMMAND STARTS WHILE LOOP

FIND FAILURE KILLS SECOND COMMAND

WHILE-BEGIN-END COMMANDS

FIRST WHILE COMMAND BEGIN ALLOWS MORE COMMANDS END STARTS WHILE LOOP

```
 /TEXT EDLAB
 /WHILE
 /   FINDQ "PROGRAMMER"
 /   BEGIN
 /      CHANGE "PROGRAMMER" TO "PGMR"
 /      HOLDQ *,APPEND
 /      END
   /FINDQ "PROGRAMMER"
   /BEGIN
   /CHANGE "PROGRAMMER" TO "PGMR"
     7      SUCCESSFUL, AND THIS COAL IS THE PRIMARY AIM OF A PGMR.
   /HOLDQ *,APPEND
 HOLD FILE LENGTH IS 1 RECORD
   /END
   /FINDQ "PROGRAMMER"
   /BEGIN
   /CHANGE "PROGRAMMER" TO "PGMR"
     9      BECOME MORE IMPORTANT AS A PGMR GAINS EXPERIENCE AND AS THE
   /HOLDQ *,APPEND
 HOLD FILE LENGTH IS 2 RECORDS
   /END
   /FINDQ "PROGRAMMER"
   /BEGIN
   /CHANGE "PROGRAMMER" TO "PGMR"
    19      PGMR MUST USE AS FEW INSTRUCTIONS AS POSSIBLE AND CHOOSE
   /HOLDQ *,APPEND
 HOLD FILE LENGTH IS 3 RECORDS
   /END
   /FINDQ "PROGRAMMER"
   /BEGIN
   /CHANGE "PROGRAMMER" TO "PGMR"
    21      CYCLES THAT ARE REQUIRED.  ALSO THE PGMR MUST USE INPUT-OUTPUT
   /HOLDQ *,APPEND
 HOLD FILE LENGTH IS 4 RECORDS
   /END
```

WHILE-BEGIN-END COMMANDS

```
  /FINDQ "PROGRAMMER"
  /BEGIN
  /CHANGE "PROGRAMMER" TO "PGMR"
   24      PGMR.  QUITE OFTEN A PROGRAMMER BECOMES SO ANXIOUS TO SAVE
  /HOLDQ *,APPEND
HOLD FILE LENGTH IS 5 RECORDS
  /END
  /FINDQ "PROGRAMMER"
  /BEGIN
  /CHANGE "PROGRAMMER" TO "PGMR"
   24      PGMR.  QUITE OFTEN A PGMR BECOMES SO ANXIOUS TO SAVE
  /HOLDQ *,APPEND
HOLD FILE LENGTH IS 6 RECORDS
  /END
  /FINDQ "PROGRAMMER"
  /BEGIN
  /CHANGE "PROGRAMMER" TO "PGMR"
   32      DESIGNS A PROGRAM, THE PGMR SHOULD TRY TO ANTICIPATE AS
  /HOLDQ *,APPEND
HOLD FILE LENGTH IS 7 RECORDS
  /END
  /FINDQ "PROGRAMMER"
  /BEGIN
  /CHANGE "PROGRAMMER" TO "PGMR"
   52      THIS SITUATION HAS OCCURRED MANY TIMES.  IT IS UP TO THE PGMR,
  /HOLDQ *,APPEND
HOLD FILE LENGTH IS 8 RECORDS
  /END
  /FINDQ "PROGRAMMER"
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
 -/BEGIN
 -/CHANGE "PROGRAMMER" TO "PGMR"
 -/HOLDQ *,APPEND
 -/END
/
```

FIND FAILURE SETS
FLAG TO FALSE-KILLS
REST OF COMMAND

WHILE-BEGIN-END COMMANDS

```
/TEXT EDLAB
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? Y
/LIST 1Ø
   1Ø     PROGRAMMING PROBLEMS BECOME MORE INVOLVED.


/FIND FIRST
     1                                        WHAT IS A GOOD PROGRAM? *
         ↑(1    )


/WHILE
/    FIND "PROGRAMMING"(+1Ø)
/    BEGIN
/        FIND "PROBLEMS"/*(+1Ø)
/        LIST *
/        END
```

(+10) SETS POINTER TO END OF STRING

/*(+10) SEARCHES NEXT 10 CHARACTERS FOR SECOND STRING

WHILE-BEGIN-END COMMAND

```
/FIND "PROGRAMMING"(+1Ø)
 1Ø      PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
                    (11  )↑                       FIRST STRING FOUND
/BEGIN
/FIND "PROBLEMS"/*(+1Ø)                           SECOND STRING FOUND
 1Ø      PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
                     (13  )↑
/LIST *                                           LIST COMMAND EXECUTES
 1Ø      PROGRAMMING PROBLEMS BECOME MORE INVOLVED.
/END
/FIND "PROGRAMMING"(+1Ø)                          FIRST STRING FOUND
 49      *WALNUT, FRANCIS K., INTRODUCTION TO COMPUTER PROGRAMMING AND
CODING,
                                                            (57  )↑

/BEGIN
/FIND "PROBLEMS"/*(+1Ø)
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 3
 -/LIST *
 -/END
  /FIND "PROGRAMMING"(+1Ø)
*21*STRING NOT FOUND BEFORE LIMIT
AT DEPTH 2
 -/BEGIN
 -/FIND "PROBLEMS"/*(+1Ø)
 -/LIST *
 -/END
 /
```

END COMMAND

E[ND]

TERMINATES TEXT EDITOR

```
/END
IF IT IS OK TO CLEAR RESPOND "YES"
CLEAR? YES

 END OF SUBSYSTEM
:
```

BASIC EDITOR

SUBSET FOR PROGRAM DEVELOPMENT

● TEXT

● ADD

● KEEP

● LIST

● CHANGE

● DELETE

● MODIFY

● FIND

● END

EDITOR REVIEW QUESTIONS

1. The Modify command is used to:

   a. add a line to a text file
   b. make corrections to a file
   c. delete lines from a text file

2. The Insert command will insert a line after the position specified.

   a. True
   b. False

3. The Add command can be used to replace an existing line in a Text File.

   a. True
   b. False

4. The ___________ command moves the Editor pointer to a specified location in the text file.

5. The ___________ command can be used to renumber lines in a text file.

6. The ___________ command is used to clear the Hold file.

7. The prompt character from the Editor is:

   a. >
   b. /
   c. -

8. The ___________ command is used to terminate the Editor subsystem.

   a. End
   b. Bye
   c. Exit
   d. a or c

9. The add command can be terminated by CNTRL Y or _______________.

   a. break
   b. double slash (//)
   c. CNTRLX

10. To use the OFFLINE parameter of the LIST command and have the listing appear on the line printer, the user must have

    a. given a FILE command for the printer and invoke the EDITOR with the listfile parameter.
    b. made sure the device is off-line.
    c. used a FILE command with EDTLIST as the formaldesignator.
    d. a and c
    e. a or c

11. The Delete command can remove both character and lines from a text tile.

    a. True
    b. False

EDITOR REVIEW QUESTIONS (CONTINUED)

12. Within the Modify command which of the following operations are valid:

    a. replace, change, insert
    b. replace, delete
    c. insert, delete, replace
    d. insert, delete
    e. insert, replace, change, delete

ANSWERS TO EDITOR REVIEW QUESTIONS

1. b
2. b, False, insert will go before position specified.
3. b, False, the add command will not replace or interleave lines
4. Find
5. Gather
6. Hold
7. b
8. d
9. b
10. e
11. a ,True
12. c

LAB #3 (EDITOR)

Read the entire lab before starting.

Given:

1. A text file named LAB3EDIT in the PUB group of your account.
2. Another small disc file called PARA1 in the PUB group of your account.

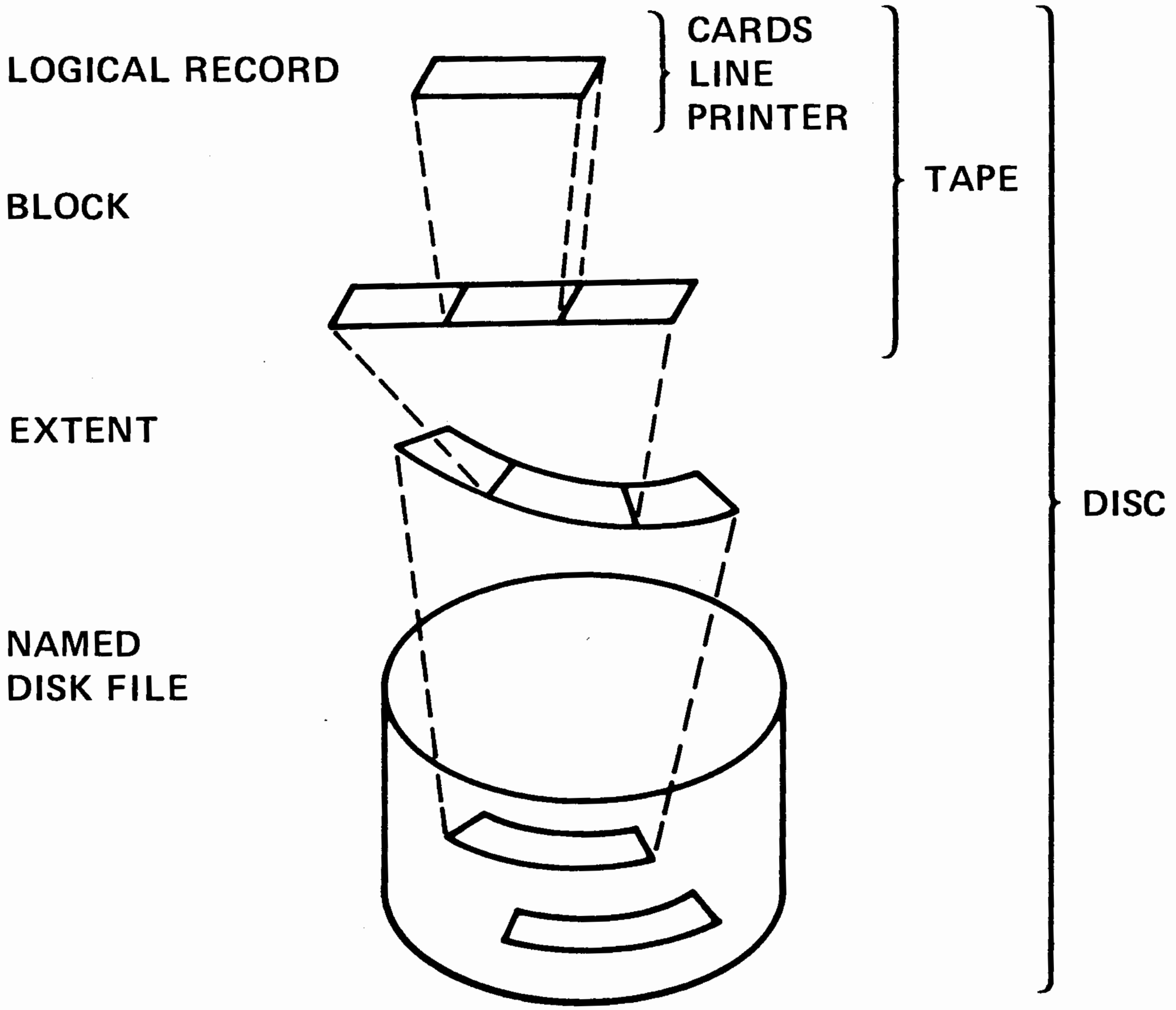
Task:

1. Log on the terminal.
2. Issue a FILE command for the line printer.
3. Invoke the Editor with the listfile parameter.
4. Read the text file into the Editor work file.
5. Obtain an offline listing of the text file.
6. Change "COAL" to "GOAL" in line 7.
7. Add "EFFICIENTLY" following line 21.
8. Insert two blank lines after line 22.
9. Insert "FOR" in front of "THEM" in line 33.
10. Insert the missing line: "FOR EXAMPLE, IN A PROGRAM PREPARED TO SOLVE THE INVENTORY PROBLEM" before line 34.
11. Change "NEGSJBIH" to "NEGATIVE" in line 41.
12. Change "T-G-G" to "STER" in line 46.
13. Delete "(DELETE)" from line 53.
14. Insert a period after "CARD" in line 57 and delete the rest of the line.
15. Add your name to line 59.
16. Paragraph #1 is missing. It is contained in the disc file PARA1.PUB. Using one Editor command insert it in front of line 16, do not affect the other existing line numbers.
17. Line 49 and 50 are out of place. With one command move them in front of line 56.
18. Renumber the file.
19. Obtain an off-line listing of the file to see your changes have taken place.
20. Save the file under the name ELAB3.
21. Exit from the Editor and Log-off.

LAB #4 (EDITOR)

Using the file EDLAB.PUB and the EDITOR, input the WHILE-FIND sequence of commands to print on the terminal, all lines in the file that have the word "computer" within 15 characters of the word "to".

FILE STRUCTURE



LOGICAL RECORD

BLOCK

EXTENT

NAMED
DISK FILE

CARDS
LINE
PRINTER

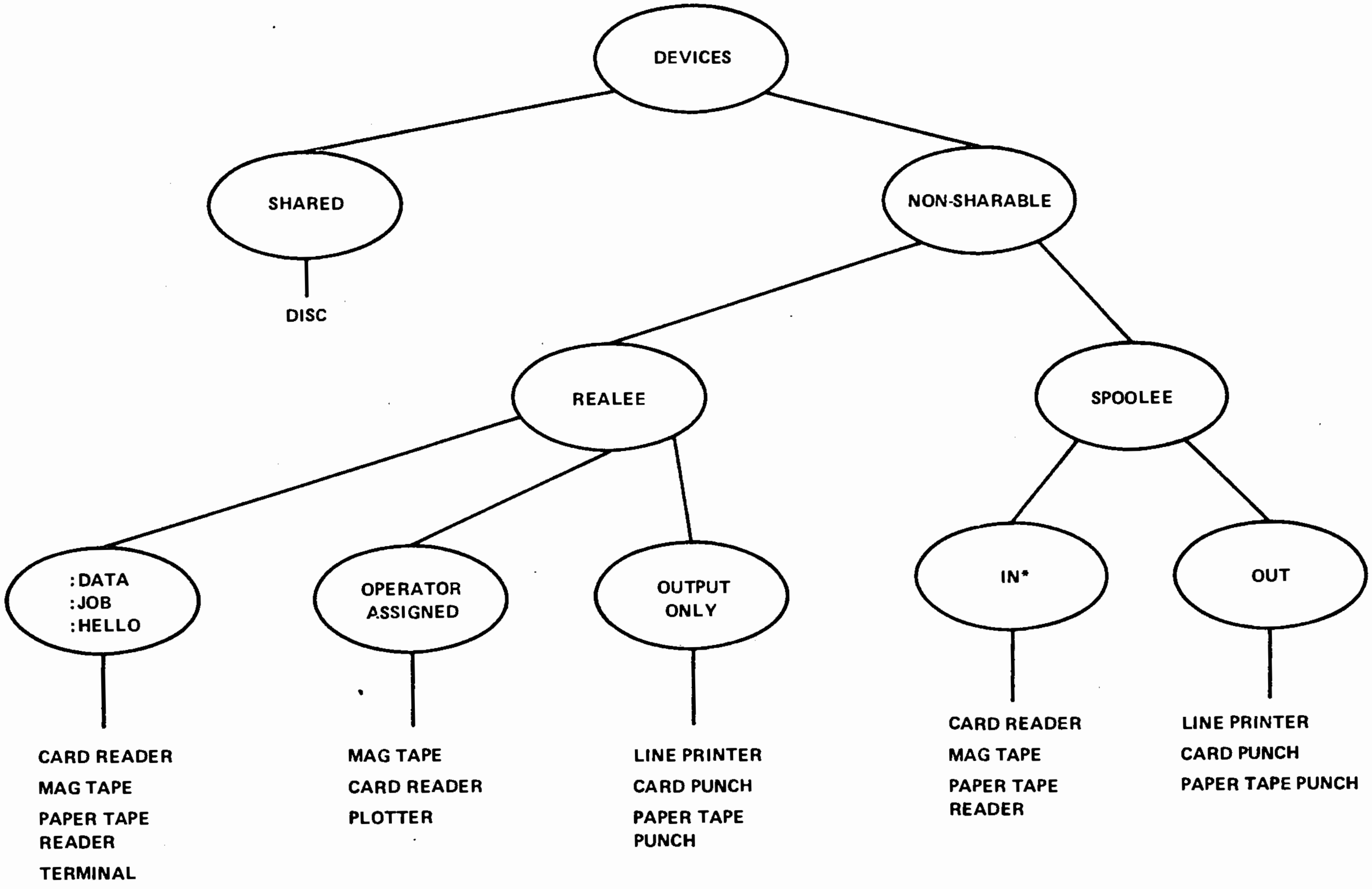
TAPE

DISC

FILES AS SEEN BY MPE

- DISC FILES
  - IMMEDIATELY ACCESSIBLE
  - POTENTIALLY SHARABLE

- DEVICEFILES
  - OWNED/ACCESSED BY JOB/SESSION
  - ANY PERIPHERAL DEVICE EXCEPT DISC



*MUST BE DATA AND/OR JOB ACCEPTING

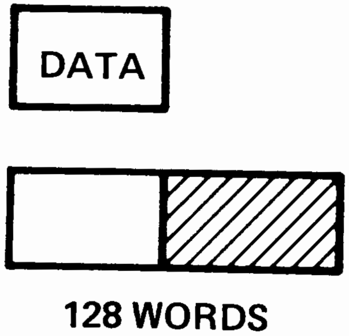
FILES AS SEEN BY MPE (CONT'D)

- DATA TYPES
  - ASCII
  - BINARY
- RECORD SIZE
  - DISC
  - TAPE
  - CONFIGURATION
- FILE FORMATS
  - FIXED
  - VARIABLE
  - UNDEFINED

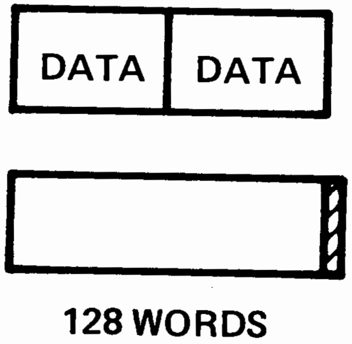
FIXED LENGTH RECORDS

BLOCK SIZE = RECORD SIZE X BLOCK FACTOR

RECORD SIZE	60 WORDS
BLOCK FACTOR	1
BLOCK SIZE	60 WORDS



RECORD SIZE	60 WORDS
BLOCK FACTOR	2
BLOCK SIZE	120 WORDS

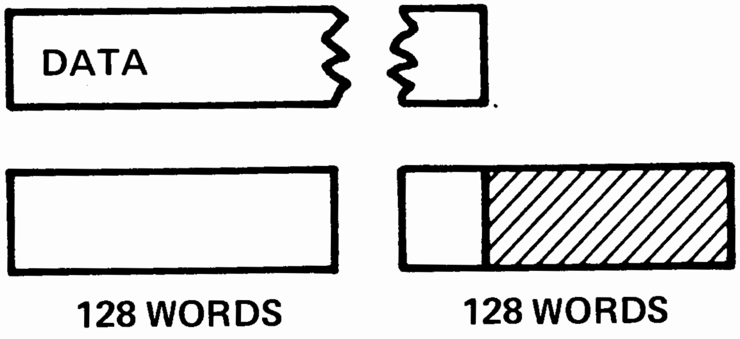


FIXED LENGTH RECORDS (CONT'D)

RECORD SIZE 160 WORDS

BLOCK FACTOR 1

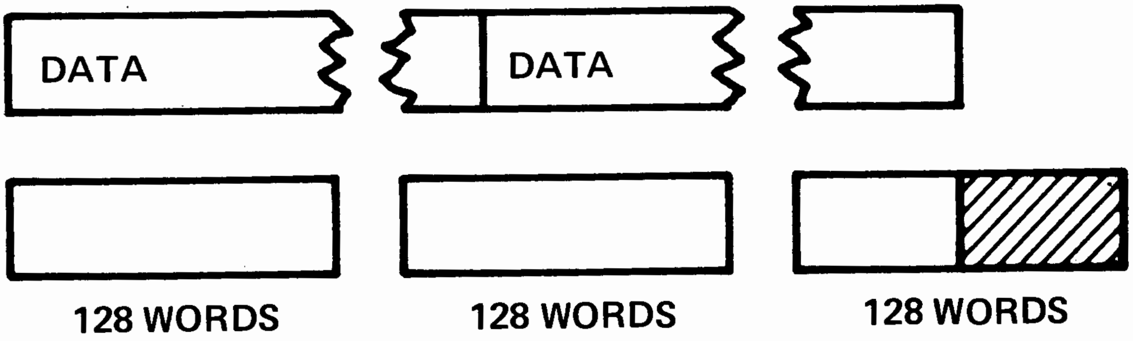
BLOCK SIZE 160 WORDS



RECORD SIZE 160 WORDS

BLOCK FACTOR 2

BLOCK SIZE 320 WORDS



BASIC PROGRAM   BLOCKOPT.PUB.SYS

```
RECORD SIZE ? 50
LOWER AND UPPER BLOCK FACTORS ? 1,30
PERCENT UTILIZATION LIST CUT OFF ? 0

BLOCK          BLOCK          LEFT OVER      PERCENT UTILIZATION
FACTOR         SIZE WORDS

 1              50             78             39.0625
 2              100            28             78.125
 3              150            106            58.5938
 4              200            56             78.125
 5              250            6              97.6563
 6              300            84             78.125
 7              350            34             91.1458
 8              400            112            78.125
 9              450            62             87.8906
 10             500            12             97.6563
 11             550            90             85.9375
 12             600            40             93.75
 13             650            118            84.6354
 14             700            68             91.1458
 15             750            18             97.6563
 16             800            96             89.2857
 17             850            46             94.8661
 18             900            124            87.8906
 19             950            74             92.7734
 20             1000           24             97.6563
 21             1050           102            91.1458
 22             1100           52             95.4861
 23             1150           2              99.8264
 24             1200           80             93.75
 25             1250           30             97.6563
 26             1300           108            92.3295
 27             1350           58             95.8807
 28             1400           8              99.4318
 29             1450           86             94.401
 30             1500           36             97.6563

RECORD SIZE ? 50
LOWER AND UPPER BLOCK FACTORS ? 1,30
PERCENT UTILIZATION LIST CUT OFF ? 99

BLOCK          BLOCK          LEFT OVER      PERCENT UTILIZATION
FACTOR         SIZE WORDS

 23             1150           2              99.8264
 28             1400           8              99.4318
```

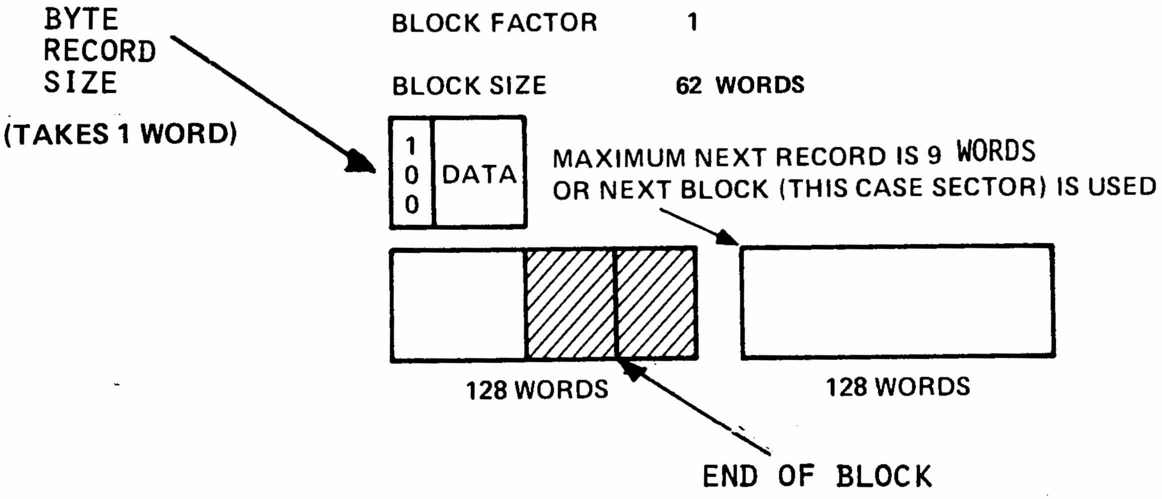
VARIABLE LENGTH RECORDS

BLOCK SIZE = RECORD SIZE X BLOCK FACTOR + 2

FOR BEST DISC UTILIZATION BLOCK SIZE
SHOULD BE A MULTIPLE OF 128

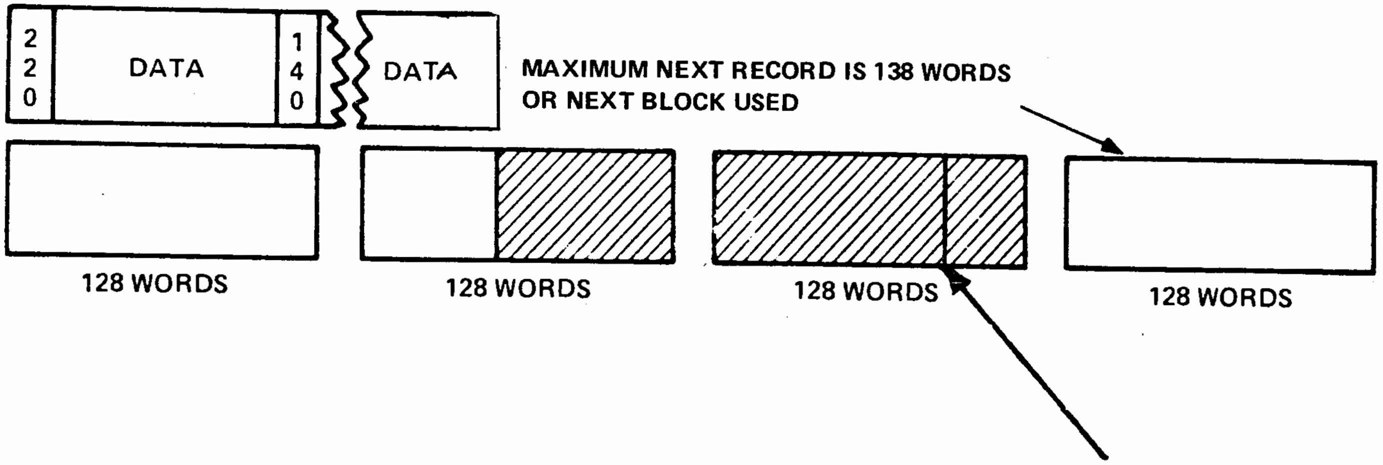
-1 RECORD SIZE INDICATES END OF BLOCK

RECORD SIZE	60 WORDS
BLOCK FACTOR	1
BLOCK SIZE	62 WORDS


BYTE
RECORD
SIZE
(TAKES 1 WORD)
1
0
0
DATA
MAXIMUM NEXT RECORD IS 9 WORDS
OR NEXT BLOCK (THIS CASE SECTOR) IS USED
128 WORDS
128 WORDS
END OF BLOCK

VARIABLE LENGTH RECORDS

RECORD SIZE	160 WORDS
BLOCK FACTOR	2
BLOCK SIZE	322 WORDS



MAXIMUM NEXT RECORD IS 138 WORDS
OR NEXT BLOCK USED

END OF BLOCK

UNDEFINED LENGTH RECORDS

- EACH RECORD IS ALWAYS 1 OR MORE SECTORS.
- USED BY COMPILERS AND OTHER SUBSYSTEMS.

FILES AS SEEN BY MPE (CONT'D)

- BLOCKING

  BLOCK FACTOR
  BLOCK SIZE = BUFFER SIZE

- NUMBER OF BUFFERS

  MAX = 16   DEFAULT = 2
  NONE (RECORD SIZE = BLOCKSIZE)

- MULTIPLE RECORDS

  INPUT
  OUTPUT

- LABEL HANDLING

  DISC
  TAPE

- PRINTER

  FORMS MESSAGE
  CARRIAGE CONTROL

SYSTEM-DEFINED (DEFAULT) FILES

$------

INPUT SET

$STDIN

$STDINX

$OLDPASS

$NULL

OUTPUT SET

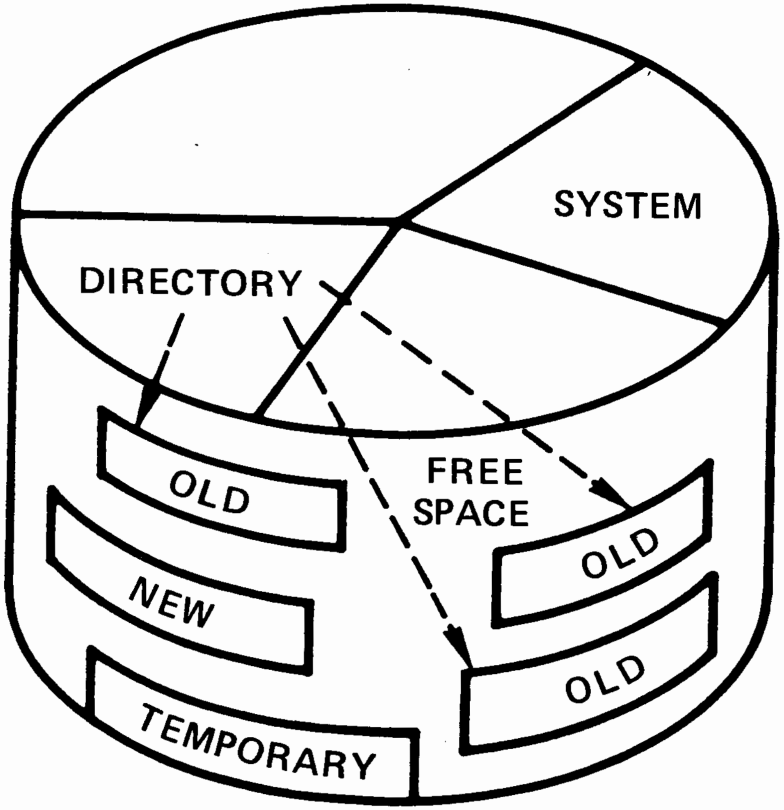
$STDLIST

$OLDPASS

$NEWPASS

$NULL

DISC USAGE AND FILE DOMAIN



DISC FILE SPACE

FREE DISC SPACE

AVAILABLE FOR USE AS FILES
EACH DISC HAS A TABLE

PROCESS INFORMATION

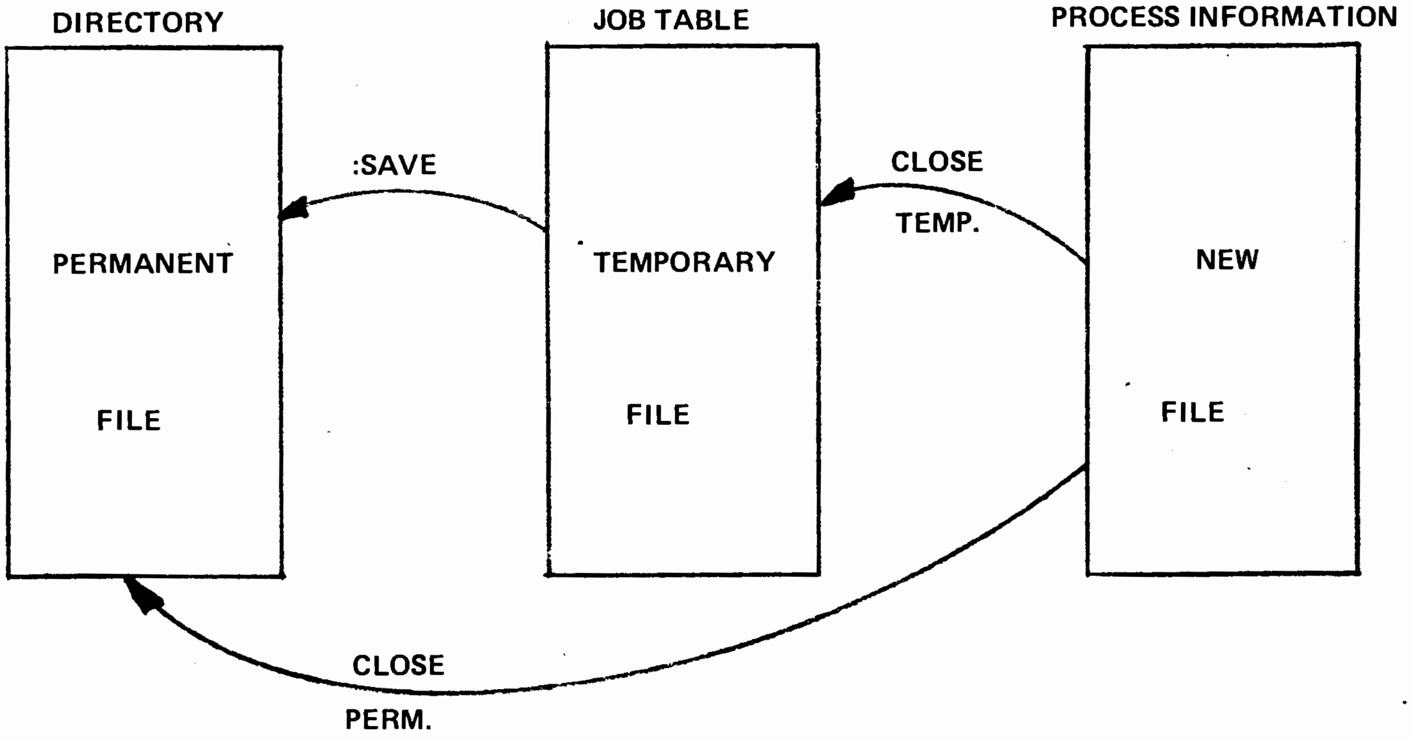
POINTS TO NEW FILES

JOB TABLES

OLD JOB TEMPORARY FILES

DIRECTORY

POINTS TO OLD PERMANENT FILES


DIRECTORY
PERMANENT
FILE
JOB TABLE
TEMPORARY
FILE
PROCESS INFORMATION
NEW
FILE
:SAVE
CLOSE
TEMP.
CLOSE
PERM.

USING FILES

or

CALLING SEQUENCE

"FOPEN"

OTHER FILE INTRINSICS

.

.

.

OTHER FILE INTRINSICS

"FCLOSE"

You do this or a subsystem does it for you

MPE FILE COMMAND
BUILD

- ALLOCATES DISC SPACE
- FILE INITIALLY EMPTY
- IMMEDIATE ALLOCATION
- PERMANENT UNLESS TEMP INDICATED

MPE "FILE" COMMANDS

```
:BUILD  filereference

     [;DEV=device]  [;DISC=[filesize] [,[numextents] [,initialloc] ]]

     [                                         [     [ F ] [             ]]]
     [;REC=[recsize] [,[blockfactor]           [  ,  [ U ] [ ,BINARY     ]]]
     [                                         [     [ V ] [ ,ASCII      ]]]

     [;CCTL  ]
     [;NOCCTL][;TEMP]  [;CODE=filecode]
```

FILE NAMES

ABCDEFGH	FILE NAME
A.PRODUCT	FILE NAME + GROUP NAME
A.PRODUCT.COMPANY	FILE NAME + GROUP NAME + ACCOUNT NAME
B/TCUDORP	FILE NAME + LOCKWORD
PAY/ROL.PERSONAL.BCOMPANY	FULLY QUALIFIED FILE NAME WITH LOCKWORD

NO PASSWORDS

35 CHARACTERS MAXIMUM FILE NAME

DEVICE PARAMETER

```
;DEV= { DEVCLASS }  - name
      { LDN      ,}  ← number (not recommended)
```

EXAMPLE:

```
;DEV  =  11
         DISC1
         DISC (DEFAULT)
```

DISC PARAMETER

```
;DISC  =  FILE SIZE,
          NUMBER OF EXTENTS,
          INITIAL EXTENT ALLOCATION

;DISC  =  10000, 10, 5
```

RECORD PARAMETER

```
;REC  =  RECORD SIZE,
         BLOCKING FACTOR,
         FIXED/UNDEFINED/VARIABLE LENGTH
         BINARY/ASCII

;REC  =  -80,3,F,ASCII
```

FILE CODES

NAME	NUMBER
USL	1Ø24
BASD	1Ø25
BASP	1Ø26
BASFP	1Ø27
RL	1Ø28
PROG	1Ø29
STAR	1Ø3Ø
SL	1Ø31
EDIT KEEPQ (NON-COBOL)	1050
EDIT KEEPQ (COBOL)	1051
EDIT (FORMAT=COBOL)	1052

< Ø	FOR PM USERS
0 – 1Ø23	FOR USER
> 1Ø23	FOR SYSTEM

MPE "FILE" COMMANDS

```
:BUILD  filereference

     [;DEV=device]  [;DISC=[filesize] [,[numextents] ·[,initialloc] ]]

                                               [   [ F ] [ ,BINARY ] ]
     [;REC=[recsize] [,[blockfactor]  [ , [ U ] [ ,ASCII  ] ]]
                                               [   [ V ] [         ] ]

     [ ;CCTL   ] [;TEMP]   [;CODE=filecode]
     [ ;NOCCTL ]

               :BUILD ANEWFILE &

               :          ;DEV=2    ;DISC=10000, 10,5 · &
               :          ;REC=-80, 3,F,ASCII         &
               :          ;CCTL     ;TEMP   ;CODE=7
```

BUILD COMMAND EXAMPLES

```
:BUILD AAUSL;CODE=USL
```

Create a permanent USL (user subprogram library) file named AAUSL. By specifying CODE=USL the record size defaults to 128, block factor 1, fixed binary file. File size defaults to 1023 records in 8 extents, 1 initially allocated.

```
:BUILD EDTPROG;CODE=PROG;DISC=500,1;TEMP
```

Create a program file named EDTPROG by specifying CODE=PROG. The record size defaults to 128 words, blocked 1, fixed binary file. The DISC= parameter specifies 500 records maximum, in one extent. A program file must reside in 1 extent. By creating the file as temporary (;TEMP) the user may issue a :SAVE command.

```
:BUILT EDTTRNS;REC=-80,16,F,ASCII;DISC=2000,10,2
```

Build a permanent user data file name EDTTRNS, 80 byte records blocked 16, fixed, ASCII. The ;DISC= parameter overrides the default file size of 1023 records. In this example the 2000 records will be spread over 10 extents (200 records each). Two extents will be initially allocated.

BUILD COMMAND EXAMPLES (CONT'D)

```
:BUILD MAILBL;DEV=DISC1;CODE=1000
```

Create a permanent disc file with a user supplied file code. The ;DEV= parameter indicates the file is to be built on a specific disc with the name DISC1.

```
:BUILD INDATA;REC=-100,6,F,ASCII;TEMP
```

Build a temporary user data file named INDATA. The record size is 100 bytes, blocked 6, fixed and ASCII.

```
:BUILD RPTFILE;REC=-132,2,F,ASCII;CCTL;DISC=1500
```

Create a disc file named RPTFILE that will be used to print a report. The data line is 132 bytes. By specifying ;CCTL an additional byte will be added to the file for the carriage control character. The ;DISC=parameter overrides the default file size of 1023 records but does take defaults on number of extents and the initial allocation.

FILE COMMAND

- ESTABLISHES EQUATION BETWEEN FORMAL AND ACTUAL DESIGNATORS
- RUN-TIME DESCRIPTION OF UNSPECIFIED FILE CHARACTERISTICS
- RUN-TIME OVERRIDE OF FILE CHARACTERISTICS

CREATIVE FILE EQUATIONS

```
:FILE QSLIST,NEW;DEV=DISC;SAVE;NOCCTL;REC=-132,1,F,ASCII
```

- PROGRAM THAT CREATES QSLIST SPECIFIED DEVICE AS LINE PRINTER WITH CCTL. QSLIST IS CREATED AS A TEMPORARY FILE.
- FILE COMMAND USED TO OVERRIDE THE "FOPEN" WHERE CREATING PROGRAM SPECIFIED CCTL AND DEVICE.
- FILE COMMAND USED TO OVERRIDE THE "FCLOSE" OF FILE AS TEMPORARY AND MAKE FILE PERMANENT.

MPE "FILE" COMMAND

```
                                     ┌= $ NEWPASS                          ┐
                                     │[= filereference]  [,NEW]            │
                                     │                                     │
                                     │ =$OLDPASS                           │
:FILE  formaldesignator              │                     ┌,OLD      ┐    │
       ────────────────              └[=filereference]     └,OLDTEMP  ┘    ┘

┌                  ┌                   ┌   ┌ F ┐┌              ┐┐┐┐
│;REC = [recsize]  │,[blockfactor]     │ , │ U ││ ,BINARY      ││││  *
│                  │                   │   │ V ││ ,ASCII       ││││
└                  └                   └   └   ┘└              ┘┘┘┘

┌; CCTL   ┐
└' NOCCTL ┘  *

┌; NOWAIT ┐
└' WAIT   ┘

┌        ⎧IN      ⎫┐
│        │OUT     │
│;ACC  = │UPDATE  │
│        ⎨OUTKEEP ⎬
│        │APPEND  │
└        ⎩INOUT   ⎭┘

┌; NOBUF                ┐
└' BUF [=numbuffers]    ┘

┌   EXC  ┐
│ ; EAR  │
└   SHR  ┘

┌   MR    ┐
│ ; NOMR  │
└         ┘

┌   DEL   ┐
│ ; SAVE  │
└   TEMP  ┘

[;DEV=[device]  [,[outputpriority]  [,numcopies] ] ] *

[;CODE=filecode] *

[;DISC=[filesize]  [,[numextents]  [,initialloc] ] ] *

┌;MULTI    ┐
└'NOMULTI  ┘
```

*does not apply to old disc files

MPE "FILE" COMMAND

```
                                 ┌= $ NEWPASS                        ┐
                                 │[= filereference] [,NEW]           │
                                 │                                   │
                                 │ =$OLDPASS                         │
:FILE     formaldesignator       │                   ┌,OLD     ┐     │
                                 └[=filereference]   └,OLDTEMP ┘     ┘

[;REC = [recsize] [,[blockfactor] [, [F] [,BINARY]]]]] *
                                     [U] [,ASCII ]
                                     [V]

[;CCTL  ] *
[;NOCCTL]

[;NOWAIT]
[;WAIT  ]

[        ⎧IN      ⎫]
[        ⎪OUT     ⎪]
[        ⎪UPDATE  ⎪]
[;ACC =  ⎨OUTKEEP ⎬]      EXAMPLE:
[        ⎪APPEND  ⎪]
[        ⎩INOUT   ⎭]      : FILE   COBLNAME=ACTLNAME,NEW  &
                          :        ;REC=,8         ;NOCCTL   &
[;NOBUF              ]    :        ;ACC=INOUT      ;NOBUFF   &
[;BUF [=numbuffers]  ]    :        ;EAR   ;MR      ;SAVE     &
                          :        ;CODE=5         ;DEV=DISC1 &
[ EXC ]                   :        ;DISC=,,5
[;EAR ]
[ SHR ]                   ERR 22, 10
                          ILLEGAL PARAMETER
[;MR  ]                   :
[;NOMR]

[ DEL  ]
[;SAVE ]
[ TEMP ]

[;DEV=[device] [,[outputpriority] [,numcopies] ]] *      *do not apply to
                                                          old disc files
[;CODE=filecode] *

[;DISC=[filesize] [,[numextents] [,initialloc] ]] *

[;MULTI  ]
[;NOMULTI]
```

FILE COMMAND EXAMPLES

`:FILE WKLYTRNS;REC=-110,10,F,ASCII;DEV=TAPE`	The file WKLYTRNS is a file that exists on magnetic tape. The record format is 110 bytes, blocked 10, fixed and ASCII.
`:FILE LIST;DEV=LP`	The file equation relates the file LIST to a line printer. The record format of 132 bytes, blocked 1, fixed and ASCII is implied by the device type.
`:FILE INCARD;DEV=CARD`	The file equation relates the file INCARD to the card reader. The record format is implied by the device type.
`:FILE SCHDMSTR;SHR`	The file SCHDMSTR resides on the disc in the log-on group and account. The SHR parameter is used to permit simultaneous shared access to the file by multiple processes.
`:FILE ACKN,NEW;REC=-128,12,F,ASCII;SAVE`	The file ACKN is being created by the currently running process. The SAVE parameter indicates that when the file is closed it is to be saved in the system file domain.
`:FILE SALEFCST,NEW;REC=-60,4,F,ASCII;DISC=2000`	The file SALEFCST is being created by a currently running process. The DISC parameter is used to override the default of 1023 records.
`:FILE RUNTOTAL;REC=-80,,F,ASCII;DISC=10,1,1;TEMP`	The file RUNTOTAL is being output by a process as a job temporary file. The DISC parameters indicate that space is only required for 10 records.

FILE COMMAND EXAMPLES CONT'D

```
:FILE RUNTOTAL,OLDTEMP;DEL
```

The file RUNTOTAL is now being accessed by the next step in the job/session. The OLDTEMP parameter indicates MPE should look in the job table area for this file. The DEL parameter indicates it will be deleted from the job table and the space returned to the free space table when closed by the process.

NOTE: A similar situation to the above two commands could be accomplished by the first process issuing the command –

```
:FILE RUNTOTAL=$NEWPASS
```

and the following step in the job or session calling for the file as –

```
:FILE RUNTOTAL=$OLDPASS;DEL
```

```
:FILE DUMPFILE;DEV=TAPE;REC=-2048,1,U
```

This file command could be used to dump a magnetic tape where the record size, etc. was unknown.

```
:FILE MTDSALES;DEV=TAPE;ACC=APPEND
```

This file equation allows the process to add data to the end of a previously created magnetic tape.

```
:FILE INFILE=EMPLOYEE.PUB.PAYROLL
```

The programmer referenced a file in the program called INFILE. It's actual name on disc is EMPLOYEE. It resides in the PUB group of the PAYROLL account.

FILE COMMAND BACK-REFERENCE

- POINTS TO PREVIOUS FILE COMMAND
- TRANSFERS FILE CHARACTERISTICS
- ASTERISK INDICATES BACK-REFERENCE
- NO OTHER PARAMETERS ALLOWED

```
:FILE NEWFLE=*DUMMY
```

BACK REFERENCE EXAMPLES

```
:FILE L; DEV=LP,,2; REC=-120

:FILE FTN06=*L

:FILE ABC=*L

:FILE OUT=*L
```

FOUR PROGRAMS IN A JOB STREAM ALL REFERENCE THE SAME TYPE OF LINE PRINTER FILE BY DIFFERENT NAMES. THE COMPLETE FILE EQUATION NEED ONLY BE WRITTEN BY THE FIRST PROGRAM. SUBSEQUENT PROGRAMS MAY ACCESS THE FILE IN THE FORMAT

```
:FILE FORMALDESIGNATOR = *FORMALDESIGNATOR
```

BACK REFERENCE EXAMPLES (CONT'D)

```
: FILE DUMMY;  DEV=LP

: FILE SOURCE; DEV=CARD


: SPL *SOURCE, , *DUMMY
```

MPE "FILE" COMMANDS

```
: FILE    formaldesignator = *formaldesignator

: FILE    formaldesignator = $NULL

                             ⎧$STDIN  ⎫
: FILE    formaldesignator = ⎨$STDINX ⎬
                             ⎩$STDLIST⎭

          [;REC = [recsize] [,[blockfactor]  [, [F]       [,BINARY]]]]
                                                [U]       [,ASCII ]
                                                [V]

          [  CCTL  ] [;ACC=accesstype]    [  NOBUF             ]
          [; NOCCTL]                      [; BUF [=numbuffers] ]

          [  EXC ]   [  MR   ]
          [; EAR ]   [; NOMR ]
          [  SHR ]
```

FILE COMMAND EXAMPLES

```
:FILE EMPRPT=$STDLIST;REC=–132,1,F,ASCII
```

In session mode this would allow a report normally output to a line printer to be reassigned to the sessions standard listing device.

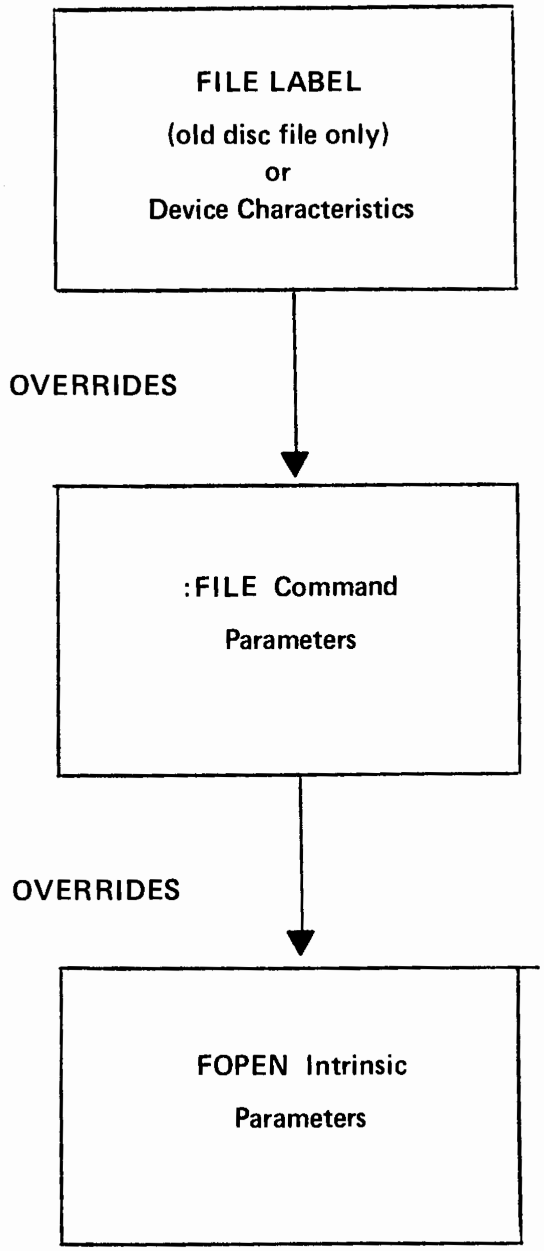
```
:FILE MASTER=$NULL
```

The $NULL parameter used with an input file generates an immediate end of file.

```
:FILE SUMMARY=$NULL
```

The $NULL parameter used with an output file will suppress the generation of the report called SUMMARY.

FILE INFORMATION HIERARCHY



FILE LABEL
(old disc file only)
or
Device Characteristics

OVERRIDES

:FILE Command
Parameters

OVERRIDES

FOPEN Intrinsic
Parameters

LAB #5 (FILES)

Task:

1. Create a disc file named LAB5 which has the following characteristics.

   A. DISC file
   B. Temporary file
   C. 4000 records
   D. No carriage control capability
   E. Ten extents total allowable
   F. One extent initally allocated
   G. 80 Byte records
   H. ASCII file
   I. Fixed length records
   J. File Code = 999

2. Make this file a permanent file in the system.

3. Change the name of the file to CHG.

4. Issue the commands to obtain all the characteristics on the printer of this and only this file with an MPE command.

5. Delete the file from the system.

6. Issue the command to see if the file still exists in the system.

LAB #6 (FILES)

OBJECTIVE: This lab will illustrate how you can:

a) Change the output format of a file
b) Direct your program to process data in a different manner than was originally coded into the program.

This can be done with FILE commands.

Given:

1. A disc file (80 byte records, ASCII) named MAILLIST in the PUB group of your account.

   a) POS 1-20 NAME
   b) POS 21-40 ADDRESS
   c) POS 41-60 CITY-STATE
   d) POS 61-80 blank

2. An SPL program callable by :RUN PRINTIT.PUB

   a) Input file formal designator is "INPUT"
   b) Output file formal designator is "OUTPUT"

Task:

Using the appropriate FILE commands and the program PRINTIT, output to your terminal the file with NAME, ADDRESS, and CITY-STATE each on a different line with a blank line following.

The output should look like this:

```
JOE BLOW
1029 Prep Lane
Cupertino, Calif.
(blank line)
WILLIE WONDER
14820 Stevens Creek
Santa Clara, Calif.
(blank line)
etc.
```

You do not need to modify the program, all necessary changes can be made with FILE commands.

Compilers & Opening Files

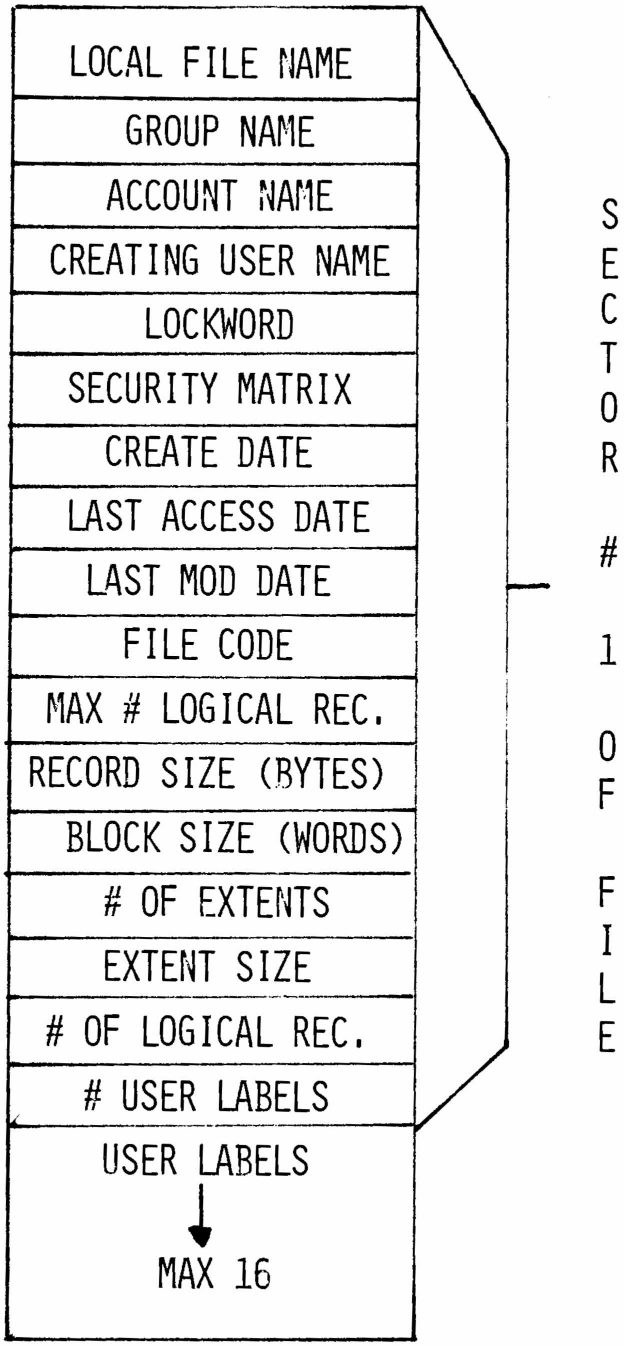
* COBOL AND RPG TRY TO OPEN FILES AS OLD IF NO SUCH FILE THE COMPILER WILL OPEN AS NEW

* FORTRAN OPENS FILES AS NEW

* BASIC ASSUMES AN OLD FILE

* THE FILE COMMAND MAY BE USED TO OVERRIDE

DISC LABEL



LOCAL FILE NAME
GROUP NAME
ACCOUNT NAME
CREATING USER NAME
LOCKWORD
SECURITY MATRIX
CREATE DATE
LAST ACCESS DATE
LAST MOD DATE
FILE CODE
MAX # LOGICAL REC.
RECORD SIZE (BYTES)
BLOCK SIZE (WORDS)
OF EXTENTS
EXTENT SIZE
OF LOGICAL REC.
USER LABELS
USER LABELS
MAX 16

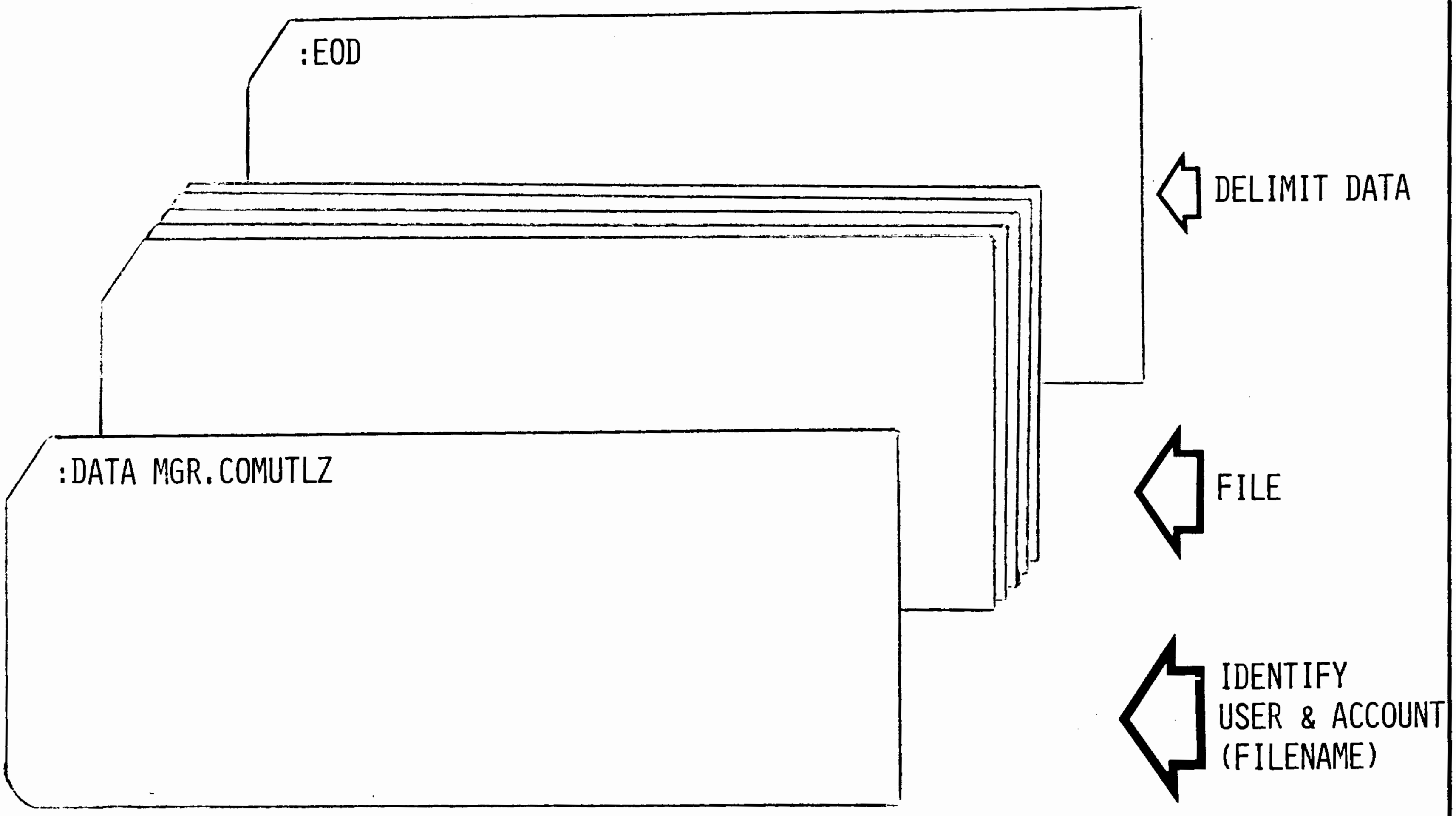
EACH USER
LABEL TAKES
ONE SECTOR

MPE 'JOB CONTROL' COMMANDS

```
:DATA  [SESSIONNAME,]  USERNAME [/UPAS] .ACCT NAME [/APAS] [;FILENAME]
       [JOBNAME,    ]
   .
   .
   .
 USER DATA
   .
   .
   .
:EOD
```

```
:DATA TX78,NEOPHYTE/IFORGOTT.COMMERCE/WHATISIT;TX78N02
```

SAMPLE DATA FILE



:EOD

DELIMIT DATA

FILE

:DATA MGR.COMUTLZ

IDENTIFY
USER & ACCOUNT
(FILENAME)

PASSWORD REQUIRED (IF APPLICABLE)

MISCELLANEOUS "FILE" COMMANDS

```
                                 printer
:LISTF [fileset] [,detail] [;listfile]

    :PURGE filereference [,TEMP]

        :RENAME  oldreference, newreference [,TEMP]

            :RESET  { @                 }
                    { formaldesignator  }

                :SAVE  { $OLDPASS, newfilereference }
                       { tempfilereference          }
```

MISCELLANEOUS COMMAND EXAMPLES

:FILE LIST;DEV=LP :LISTF @.PUB.SYS;*LIST	Commands used to list the filenames of all the files in the PUB group of the SYS account, on the line printer.
:PURGE X	Purge the permanent file X from the system.
:PURGE Y,TEMP	Purge the temporary file Y from the system.
:RENAME TRANSIN,TRANSOK/OUT.MARCH	Rename the permanent file TRANIN to TRANSOK. TRANSIN will no longer exist. A group name may be specified if you have save file access in that group (MARCH). The rename may also be used to supply a lockword (OUT)
:RENAME DFILE,WORKFL,TEMP	Rename the temporary file DFILE to WORKFL. WORKFL will remain as a temporary file.
:RESET LIST	Nullify the previously issued file command for the file named LIST.
:SAVE WORKFL	WORKFL will be made a permanent file.
:SAVE $OLDPASS,EDPROG	Make the contents of $OLDPASS permanent under the name EDPROG

LISTEQ CAPABILITIES

- LIST TEMPORARY FILES
  - LIST FILE EQUATIONS

```
:RUN LISTEQ2.PUB.SYS

***TEMP FILES

TEMP1.COBOL.TRAINING
BBBB.COBOL.TRAINING
T6781W.COBOL.TRAINING

***FILE EQUATIONS

:FILE  LP;DEV=LP
:FILE  C;DEV=CARD
:FILE  T;DEV=TAPE;REC=80,10,F,ASCII
:FILE  TAPE;DEV=TAPE;REC=80,1,V,BINARY
:FILE  LISTER=*LP

END OF PROGRAM
:
```

MISCELLANEOUS "FILE" COMMANDS

: RELEASE filereference

: SECURE filereference

:ALTSEC filereference [; ([modelist:userlist[;modelist:userlist] . . .])]

MISCELLANEOUS FILE COMMANDS EXAMPLES

:RELEASE PRODATA	This command will temporarily suspend the file security on the file named PRODATA. This will allow it to be accessed in any fashion by any user.
:SECURE PRODATA	This command will restore the security on PRODATA suspended by the :RELEASE command.
:ALTSEC PRODATA (R:ANY;W:GL,CR)	This command will alter the security on PRODATA to allow reading by any user and write access* by a group librarian or file creator. *Write access implies lock and append access.
:ALTSEC PRODATA	This command will restore default security on PRODATA.

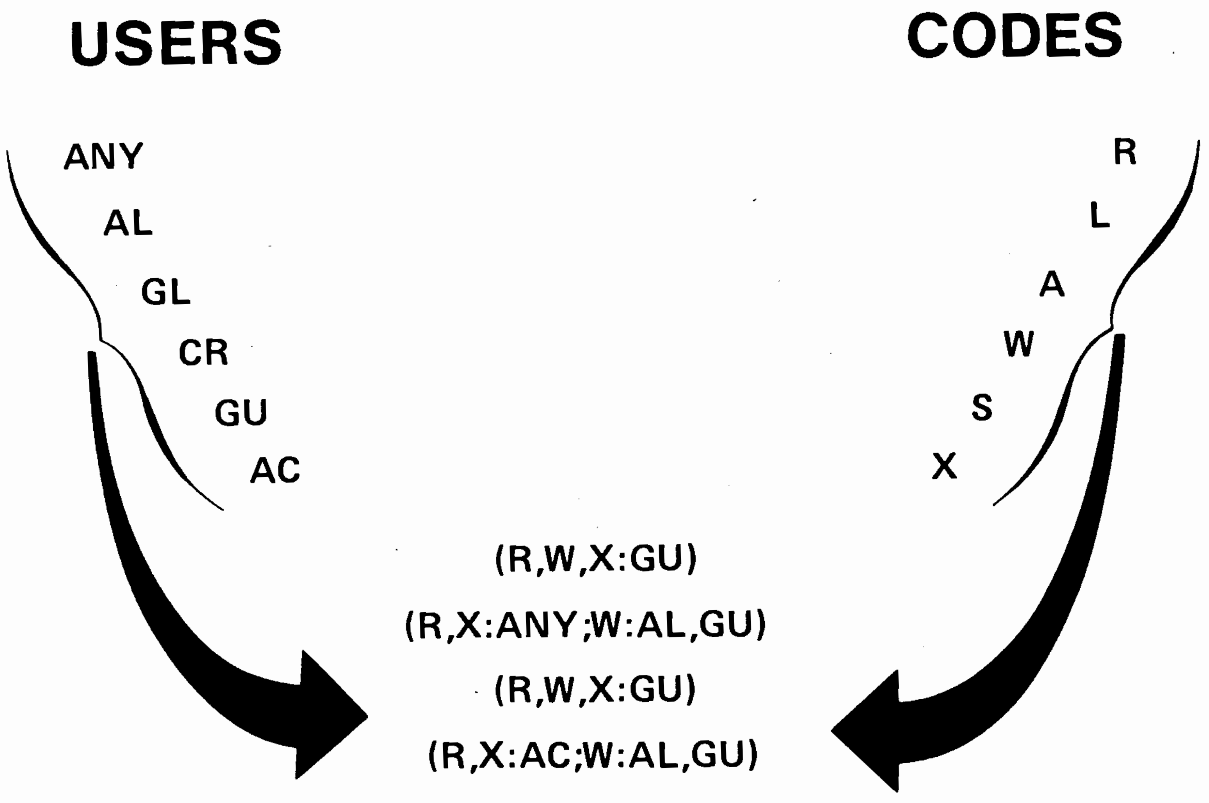
FILE SECURITY RULES

- USERS CAN CREATE FILES ONLY IN THEIR ACCOUNT
- ONLY CR CAN MODIFY A FILE'S SECURITY
- IF LOCKWORD IS PRESENT IT IS ALWAYS REQUIRED
- AM HAVE UNLIMITED FILE ACCESS TO THEIR FILE ACCOUNTS
- SM HAS UNLIMITED FILE ACCESS BUT CAN SAVE ONLY IN SYS ACCOUNT
- RELEASE ALLOWS UNLIMITED FILE ACCESS
- RELEASE DOES NOT MODIFY FILE SECURITY SETTINGS

LEVELS OF FILE SECURITY

- ◆ ACCOUNT
- ◆ GROUP
- ◆ FILE

FILE LEVEL SECURITY



USERS

ANY
AL
GL
CR
GU
AC

CODES

R
L
A
W
S
X

(R,W,X:GU)
(R,X:ANY;W:AL,GU)
(R,W,X:GU)
(R,X:AC;W:AL,GU)

LEVELS OF FILE SECURITY (CONT'D)

STANDARD USER DEFAULT ACCESS

◇ UNLIMITED ACCESS

- ■ ALL FILES IN LOGON GROUP
- ■ ALL FILES IN HOME GROUP

◇ R AND X ACCESS ONLY

- ■ ALL FILES IN HIS PUB GROUP
- ■ ALL FILES IN PUB. SYS

◇ NO ACCESS

- ■ ALL OTHER FILES IN SYSTEM

DEFAULT FILE SECURITY SETTINGS

* ACCOUNTS

(R,X,W,A,L:AC)

* GROUPS

(R,X,S,W,A,L:GU)

* FILES

(R,X,W,A,L:ANY)

WORK EXERCISE FILE SECURITY

GIVEN:

a) User name JEAN, in the FORECAST group of the SALES account.

b) Default security in effect on the system.

Do you have access to the files used in the following commands? If no, why not?

```
1.   :FILE ABC.PUB.SYS;ACC=OUT
2.   :FILE BUDGET.FINANCE.INFOSYS,NEW;SAVE
3.   :FILE DSKIN.FORECAST.MKTG;EXC;NOBUF
4.   :FILE TRANS.PUB;ACC=IN
5.   :SAVE LAB2EDIT.PUB
6.   :RUN CPROG.FINANCE.INFOSYS
```

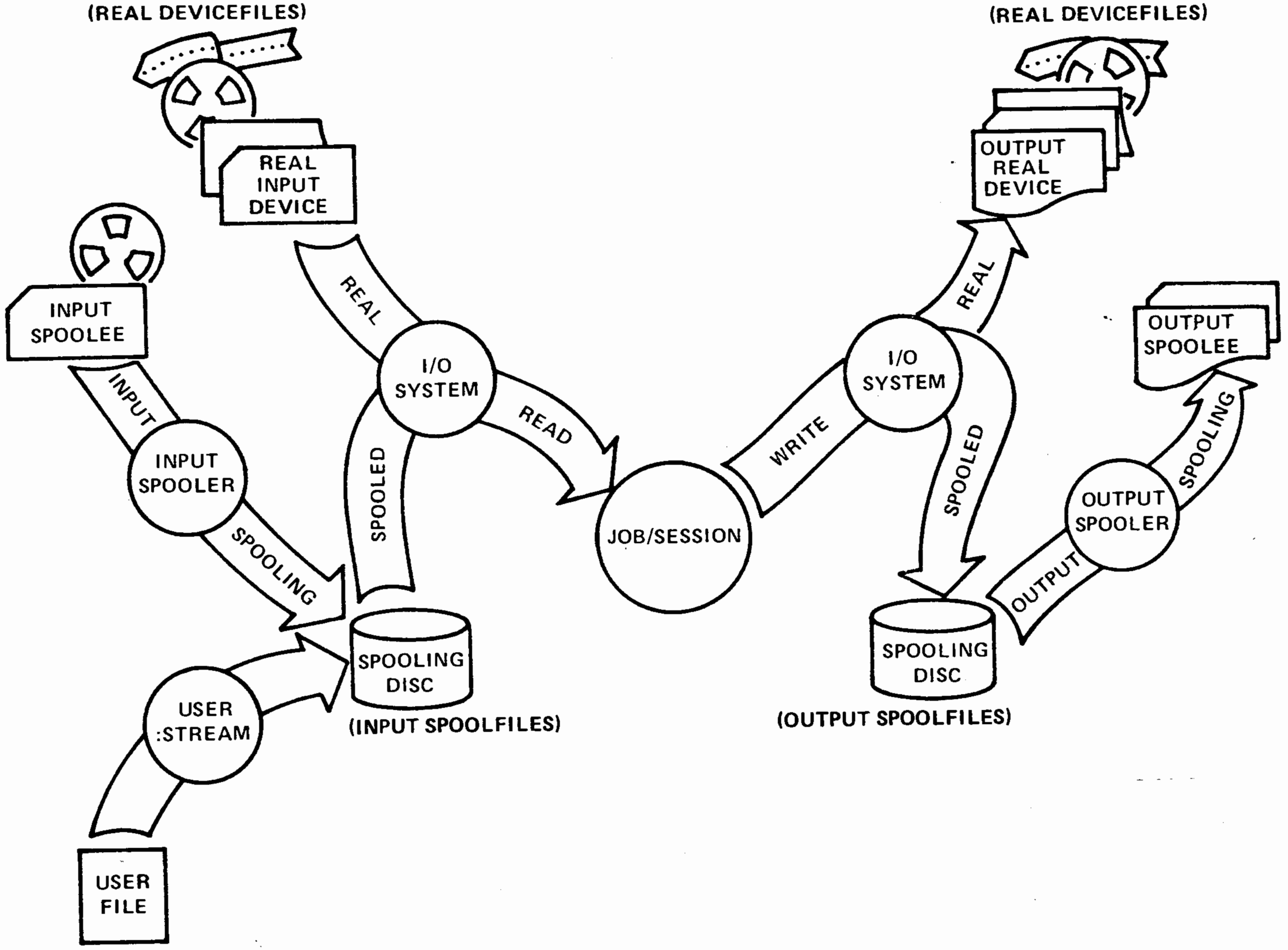
ANSWERS TO FILE SECURITY WORK EXERCISE

1. No, you do not have write access to PUB.SYS, only read and execute access.
2. No, you do not have any access to files outside your account except PUB.SYS
3. No, the file must be in the log-on group of the log-on account. MKTG was not your log-on account.
4. Yes, your log-on account is implied. You do have read access to all files in the PUB group of your log-on account.
5. No, you do not have save file access in the PUB group of your account.
6. No, you do not have execute access outside your log-on account, except in PUB.SYS

SPOOLING

- BETTER SCHEDULING
  - allows priority I/O
  - allows deferred I/O
- CONTROLLED ON DEVICE BASIS
  - reduces contention
  - better utilization
- CONSOLE OPERATOR CONTROLLED
  - transparent to the USER

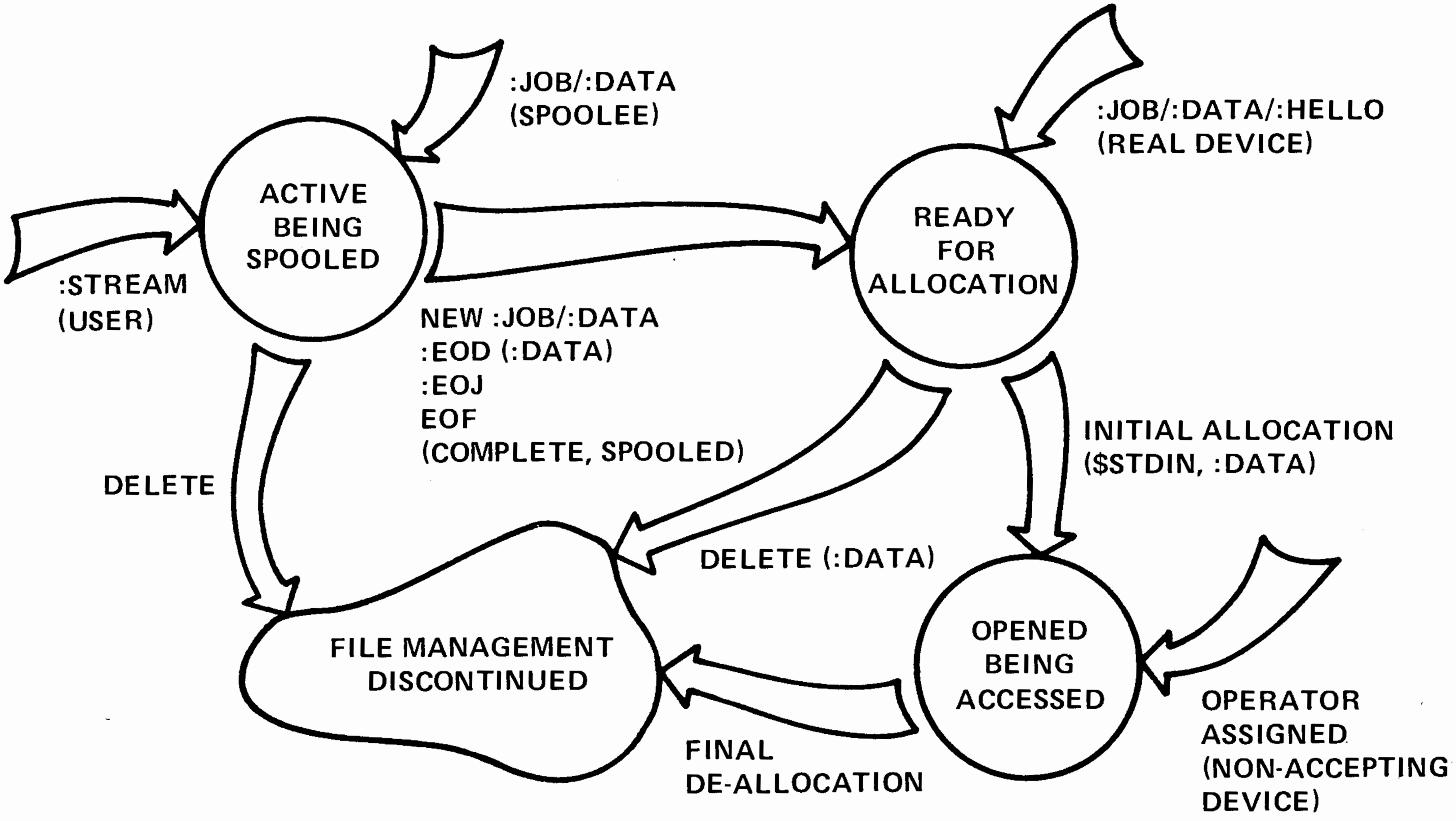
DEVICEFILE ACCESS



SPOOLING TERMINOLOGY

- DEVICEFILE
- SPOOLFILE
- SPOOLER
- SPOOLEE
- ACTIVE
- READY
- OPENED

DEVICEFILE STATES (INPUT)



:JOB/:DATA
(SPOOLEE)

:JOB/:DATA/:HELLO
(REAL DEVICE)

ACTIVE
BEING
SPOOLED

:STREAM
(USER)

READY
FOR
ALLOCATION

NEW :JOB/:DATA
:EOD (:DATA)
:EOJ
EOF
(COMPLETE, SPOOLED)

INITIAL ALLOCATION
($STDIN, :DATA)

DELETE

DELETE (:DATA)

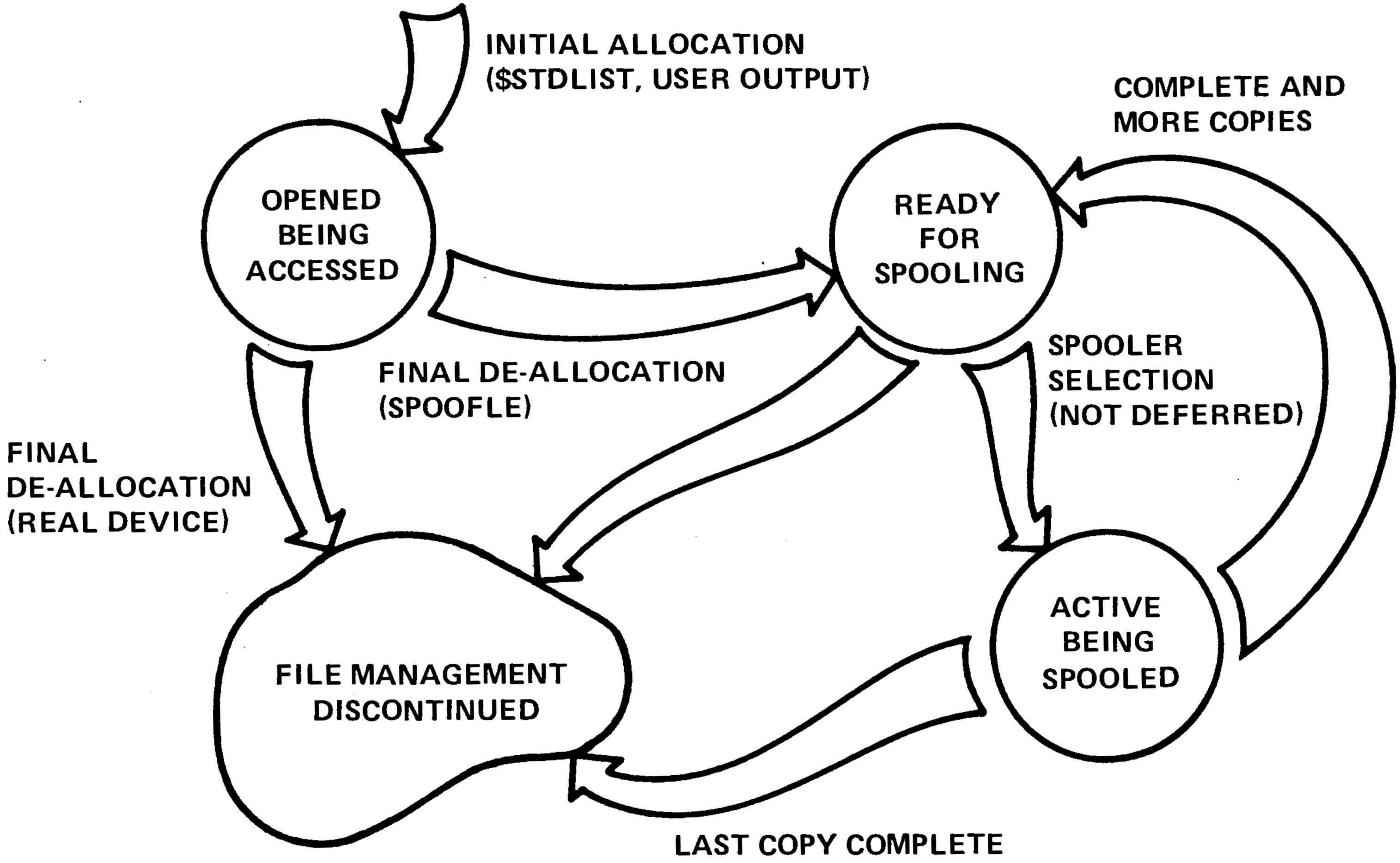
OPENED
BEING
ACCESSED

FILE MANAGEMENT
DISCONTINUED

OPERATOR
ASSIGNED
(NON-ACCEPTING
DEVICE)

FINAL
DE-ALLOCATION

DEVICEFILE STATES (OUTPUT)



STREAM COMMAND

TO INTRODUCE A JOB TO MPE/3000 FOR SCHEDULING - INDEPENDENT OF ORIGINATING JOB

```
:STREAM  [FILEDESIGNATOR] [,CHARACTER]
```

DURING A SESSION:

```
:STREAM
>!JOB MGR.COMUTLZ
>!COBOL SFILE
>!PREP $OLDPASS,PFILE
>!RUN PFILE
>!SAVE PFILE                    {JOB NUMBER RETURNED
>!EOJ                           {BY SYSTEM    #JNNN
>:EOD                    END OF STREAM INDICATED BY :EOD
```

DURING A JOB:

```
:JOB EDIT,MGR.COMULTZ
:RUN EDITPGM
:STREAM,!
!JOB RPTERR,MGR.COMUTLZ
!RUN ERROR
!EOJ
:EOD
:RUN RPT
:EOJ
```

```
PAGE 1  HEWLETT-PACKARD 32212A.1.03 FILE COPIER  THU, JUL 1

SYNTAX ERROR: EOF FOUND IN COMMAND FILE, MISSING EXIT COMMA


ERR 2 ABNORMAL PROGRAM TERMINATION
PREMATURE JOB TERMINATION

CPU (SEC) = 1
ELAPSED (MIN) = 1
THU, JUL 14, 1977,   4:23 PM
END OF JOB
```

ND

STREAM (CONT'D)

USING THE 3000/EDITOR

```
:EDITOR
/A     1      !JOB MGR.COMUTLZ
       2      !COMMENT WEEKLY JOB STREAM
       3      !FILE REM3001=ORDFILE1
       4      !PURGE ACKFILE1
       5      !BUILD ACKFILE1;REC=80,1,F,ASCII;DISC=2500
       6      !FILE REM3002=ACKFILE1
       7      !RUN REM3000P;LIB=G
       8      !RUN DBSTORE.PUB.SYS
       9      REMDBI
      10      !EOD
      11      !COMMENT  END OF WEEKLY JOB STREAM
      12      !EOJ
      13      //

/ K REM300J,UNN
/ E

      END OF SUBSYSTEM

: STREAM REM300J
  #J25
```

STREAM FILES CREATED WITH THE EDITOR MUST BE KEPT UNNUMBERED.

CAN USE :SHOWJOB TO DETERMINE JOB STATUS

STREAM (CONTINUED)

NESTED :STREAMS

WHEN ONE :STREAMED JOB IS INTRODUCED FROM WITHIN ANOTHER :STREAMED JOB, THEY MUST USE DIFFERENT PROMPT CHARACTERS.

```
:JOB START,MGR.COMUTLZ                          *OPERATOR WILL BE
:FILE T=START;DEV=TAPE                           PROMPTED FOR TAPE
:RESTORE *T;@.G1,@.G3;SHOW                       "START"
:STREAM
!JOB REPORT1,MGR.COMUTLZ
!RUN VALRRT
!STREAM ,#
#JOB ERRRPT1,MGR.COMUTLZ
#FILE P;DEV=PUNCH
#RUN ERRRPT1
#EOJ
!FILE T=CORPTAPE;DEV=TAPE;REC=-50,10,F,ASCII
!RUN FCOPY.PUB.SYS                                *OPERATOR WILL BE
FROM=CORPTAPE;TO=*T;EBCDICOUT;VERIFY               PROMPTED FOR "CORPTAPE"
EXIT
!EOJ
!EOD
!JOB REPORT2,MGR.COMUTLZ
!RUN TRANSUM
!STREAM ,#
#JOB ERRRPT2,MGR.COMUTLZ                          *OPERATOR WILL BE
#FILE RPTERR2;DEV=TAPE                             PROMPTED FOR "RPTERR2"
#RUN ERRRPT2
#EOJ
!EOD
!EOJ
:EOD
:TELLOP  WEEKLY REPORT JOBS STARTED
:EOJ
```

NOTE: USEFUL TO ENSURE JOB'S INITIATED SEQUENTIALLY

DESCRIPTION OF JOB STREAM

1. RESTORE ALL FILES IN GROUPS G1 AND G3 IN THE LOG-ON GROUP.

2. START A SERIES OF JOBS THAT WILL CREATE REPORTS FROM FILES JUST RESTORED.

3. THE PROGRAM VALRPT WILL CREATE A FILE THAT IS TO BE USED TO PUNCH ERROR RECORDS. WHEN VALRPT IS FINISHED THE JOB TO PUNCH THESE RECORDS IS STARTED.

4. AFTER ERRRPT1 HAS BEEN STREAMED, FCOPY IS USED TO COPY CORPFILE TO TAPE, CONVERT TO EBCDIC AND VERIFY THE TAPE.

5. THE JOB TITLED REPORT2 WILL PRINT A TRANSACTION SUMMARY REPORT AND CREATE ANOTHER FILE OF ERRORS. THE LAST STREAMED JOB (ERRRPT2) WILL REFORMAT THESE ERROR RECORDS ONTO MAGNETIC TAPE.

6. THE TELLOP COMMAND IS USED TO INFORM THE OPERATOR THAT THE ENTIRE WEEKLY JOB STREAM HAS BEEN STARTED.

STORE / RESTORE CAPABILITIES

FILE SELECTION BY:

- WHOLE SYSTEM
  - WHOLE ACCOUNT
    - WHOLE GROUP
      - INDIVIDUAL FILES
        - OTHER ACCOUNTS
          - OTHER GROUPS
            - OTHER INDIVIDUAL FILES

depending on USER capability

➡ ALLOWS BACK-UP STORAGE

➡ ALLOWS FILE TRANSFER BETWEEN MACHINES

STORE/RESTORE

A FILE COMMAND FOR MAG TAPE IS REQUIRED

```
:STORE [FILESETLIST] ;TAPEFILE [;SHOW] [;FILES=MAXFILES]
  FILESETLIST ──→ [FILESET [,FILESET]. . .]  (DEFAULT=@)
  TAPEFILE    ──→ *FORMALDESIGNATOR   (MUST BE MAG TAPE)
  SHOW        ──→ LISTS NAMES & COUNT OF STORED FILES
  FILES       ──→ MAXIMUM NUMBER OF FILES THAT MAY BE STORED
```

```
:FILE T1;DEV=TAPE
 :STORE DBMSLABS,DEEPSC,UTLZ;*T1;SHOW
   FILES STORED = 3
     FILE    .GROUP .ACCOUNT  LDN  ADDRESS

     DBMSLABS.PUB   .JOHNSON   2   %13636
     DEEPSC  .PUB   .JOHNSON   2   %112722
     UTLZ    .PUB   .JOHNSON   2   %111Ø16
   FILES NOT STORED=Ø
```

STORE/RESTORE

A FILE COMMAND FOR MAG TAPE IS REQUIRED

```
:RESTORE TAPEFILE [;[FILESETLIST] [;KEEP] [;DEV=DEVICE] [;SHOW] [;FILES=MAXFILES]]

TAPEFILE    → *FORMALDESIGNATOR
FILESETLIST → [FILESET[,FILESET]. . .]    (DEFAULT = @.@.@)
KEEP        → IF IDENTICAL FILENAMES - KEEP THE DISC FILE
DEVICE      → LOGICAL DEVICE NAME OR NUMBER FOR RESTORE (ALL FILES)
SHOW        → SAME AS STORE
FILES       → SAME AS STORE
```

```
:FILE T1;DEV=TAPE
:RESTORE *T1;;KEEP;SHOW
  FILES RESTORED = Ø
  FILES NOT RESTORED = 3
     FILE      .GROUP  .ACCOUNT  FILESET     REASON
     DBMSLABS.PUB      .JOHNSON     1     ALREADY EXISTS
     DEEPSC   .PUB     .JOHNSON     1     ALREADY EXISTS
     UTLZ     .PUB     .JOHNSON     1     ALREADY EXISTS
```

SYSTEM CONSOLE LOG

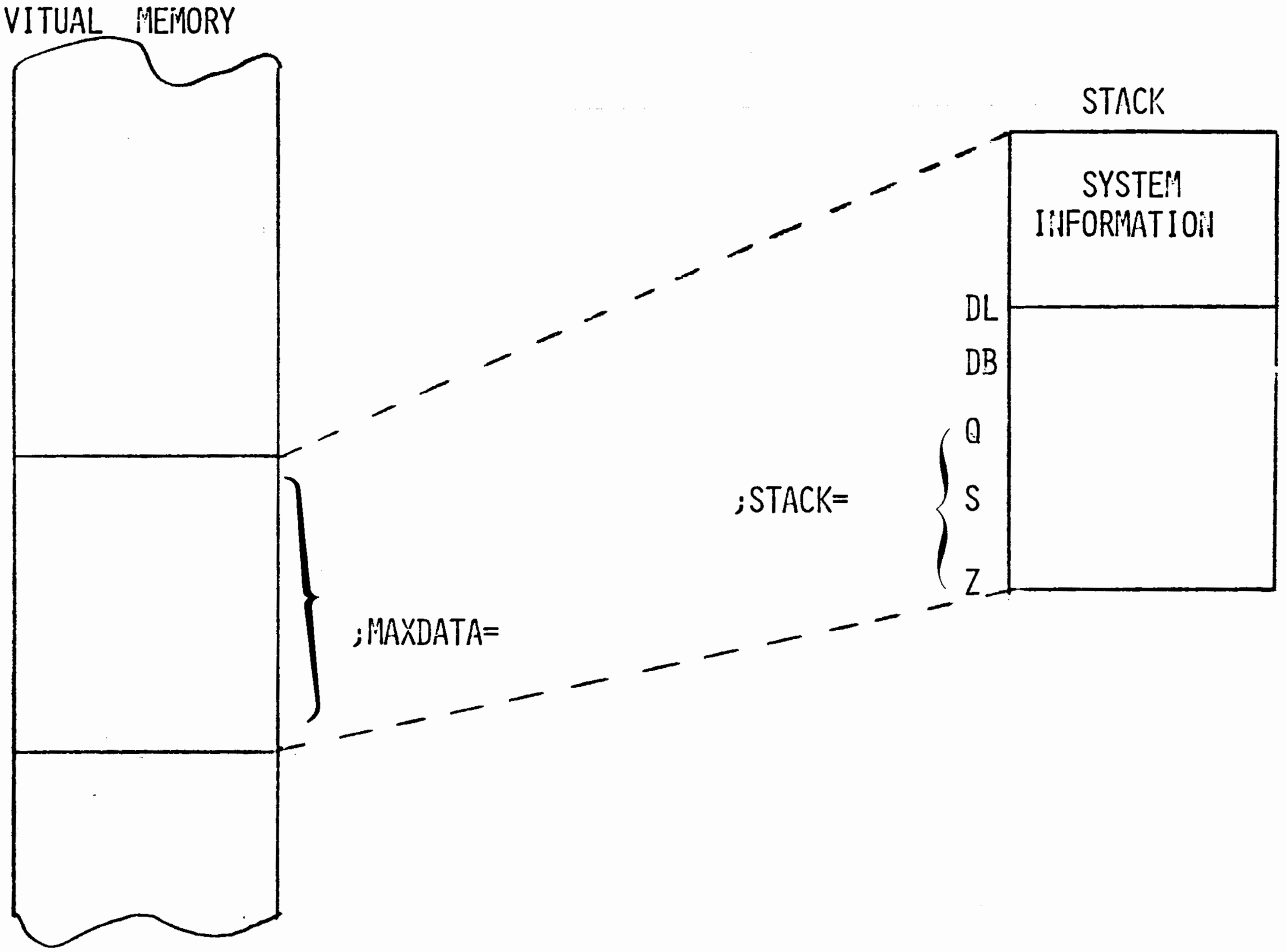
```
 ST/13:17/ILLEGITIMATE ACCESS ON LDEV#22 (COR.SCRATCH)←User tried to log on but was
 ST/13:17/#S163/LOGON FOR: USER.ACCOUNT1 ON LDEV#22         rejected because of
 ST/13:19/#S91/LOGOFF                                       illegal account
 ST/13:21/#S164/LOGON FOR: OCC.ACC ON LDEV#34
?10/13:22/#S167/40/LDEV# FOR "JCH" ON TAPE (NUM)←request to allocate tape unit
 ST/13:23/#S165/LOGON FOR: MGR.CTC ON LDEV#21
=REPLY 40,7←reply to tape allocation request

?10/13:26/#S167/40/LDEV# FOR "JCH" ON TAPE (NUM)
=REPLY 40,7
 ST/13:26/#S165/LOGOFF
 ST/13:29/#S167/LOGOFF
 ST/13:33/ILLEGITIMATE ACCESS ON LDEV#24 (MANAGER.SUPPORT)
 ST/13:33/ILLEGITIMATE ACCESS ON LDEV#24 (MANAGER.SUPPORT)
 ST/13:33/ILLEGITIMATE ACCESS ON LDEV#24 (MANAGER.SYS)
 ST/13:33/ILLEGITIMATE ACCESS ON LDEV#24 (MGR.SUPPORT)
 ST/13:35/#S167/LOGON FOR: USER.ACCOUNT1 ON LDEV#24
?10/13:36/#S167/42/LDEV# FOR "MASTER" ON TAPE (NUM)
=REPLY 42,7
?10/13:37/#S167/42/LDEV# FOR "COPY1" ON TAPE (NUM)
=REPLY,42,7
**INVALID**
=REPLY 42,7
 ST/13:39/#S167/LOGOFF
 ST/14:10/#S168/LOGON FOR: OP.SUPPLY ON LDEV#21
 10/14:13/LDEV#5 NOT READY ←card reader is empty

?10/16:02/#S110/34/LDEV# FOR "SAVE" ON TAPE (NUM)
 ST/16:02/#S112/LOGON FOR: MANAGER.SYS ON LDEV#24
 ST/16:03/#S113/LOGON FOR: MANAGER.SCP ON LDEV#21
=REPLY 34,7
=SHOWIN

DEV/CL    DFID     JOBNUM   FNAME     STATE FRM SPACE RANK PRI #C

21        #I192    #S113    $STDIN    OPENED
 ST/16:04/#S113/LOGOFF
22        #I189    #S110    $STDIN    OPENED
24        #I191    #S112    $STDIN    OPENED
33        #I163    #S91     $STDIN    OPENED
34        #I190    #S114    $STDIN    OPENED
35        #I194    #S111    $STDIN    OPENED
44        #I146    #S76     $STDIN    OPENED

7 FILES:
    0 ACTIVE
    0 READY; INCL 0 SPOOFLES, 0 DEFERRED
    7 OPENED; INCL 0 SPOOFLES
    0 SPOOFLES: 0 SECTORS
```

MPE COMMANDS

```
:RUN  progfile [,entrypoint]

        [;NOPRIV] [;LMAP] [;DEBUG] [;MAXDATA = segsize]

        [;PARM = parameternum] [;STACK = stacksize]

        [;DL = dℓ/db size] [;LIB = library] [;NOCB]


          :RUN DBUTIL.PUB.SYS,CREATE
```

MPE COMMANDS

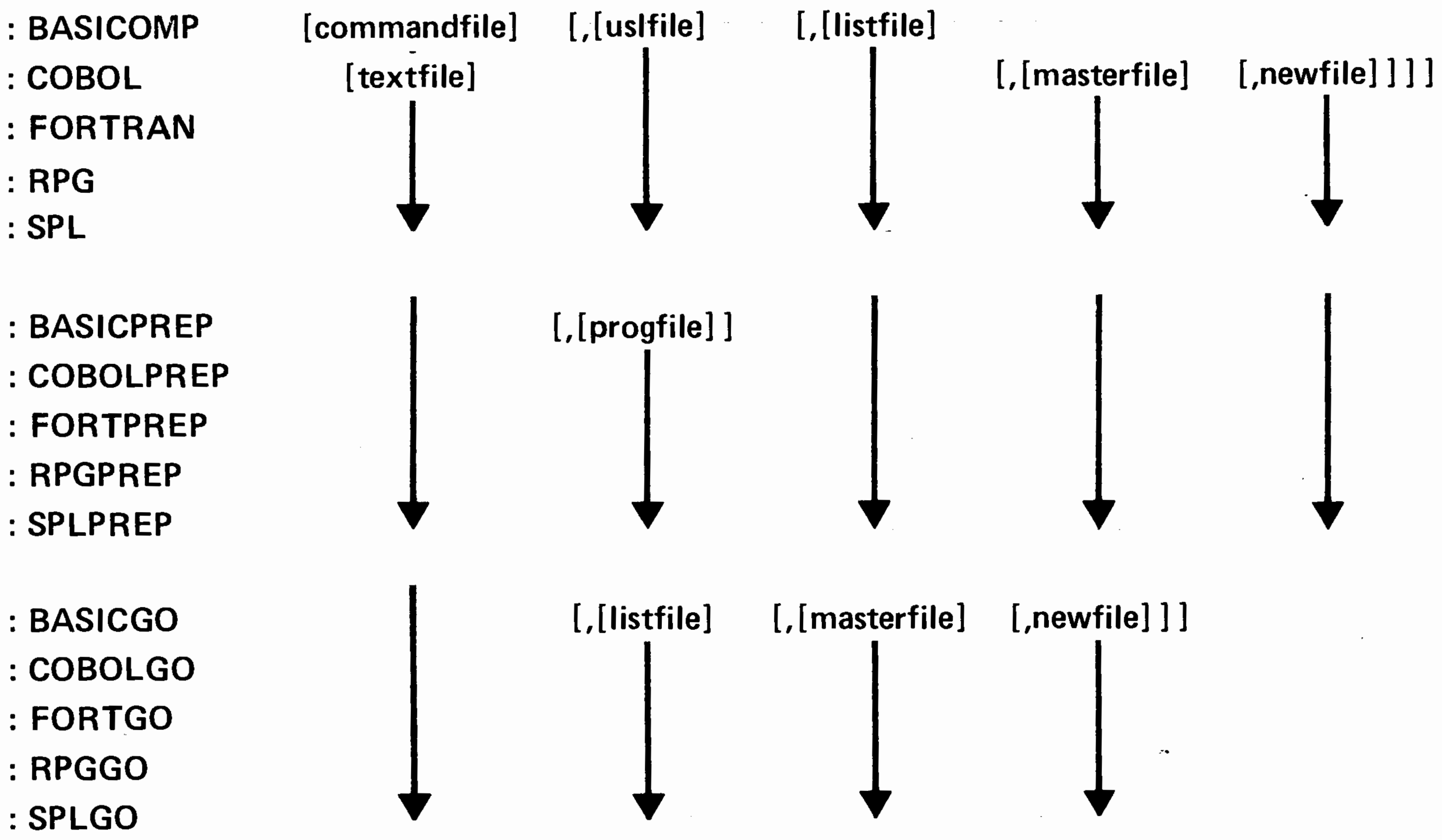
```
:PREP      uslfile, progfile

           [;ZERODB]  [;PMAP]  [;MAXDATA=segsize]

           [;STACK=stacksize]  [;DL=dℓ/db size]  [;RL=filename]


:PREP RBMODULE, RUNPROG ;RL=RELOCLIB    &
:       ;PMAP     ;MAXDATA=2ØØØØ
```


VITUAL MEMORY
;MAXDATA=
;STACK=
STACK
SYSTEM INFORMATION
DL
DB
Q
S
Z

MPE COMMANDS

```
:PREPRUN     uslfile [,entrypoint]

             [;NOPRIV] [;PMAP] [;DEBUG] [;LMAP] [;ZERODB]

             [;MAXDATA=segsize] [;PARM=parameternum]

             [;STACK=stacksize] [;DL=dℓ/db size] [;LIB=library]

             [;CAP=caplist] [;RL=filename] [;NOCB]
```

MPE "COMPILER" COMMANDS



```
: BASICCOMP     [commandfile]  [,[uslfile]    [,[listfile]
: COBOL           [textfile]                                 [,[masterfile]  [,newfile] ] ] ]
: FORTRAN
: RPG
: SPL

: BASICPREP                    [,[progfile] ]
: COBOLPREP
: FORTPREP
: RPGPREP
: SPLPREP

: BASICGO                      [,[listfile]   [,[masterfile]  [,newfile] ] ]
: COBOLGO
: FORTGO
: RPGGO
: SPLGO
```

FILE SYSTEM REVIEW QUESTIONS

1. Disk Files cannot cross volumes.

   a. True
   b. False

2. $STDLIST is always a line printer.

   a. True
   b. False

3. To suppress a compilation listing the user could specify $NULL as the listfile.

   a. True
   b. False

4. The Build command always builds a temporary file.

   a. True
   b. False

5. Lockword may be specified when using the Build command.

   a. True
   b. False

6. A disk file is made up of

   a. Segments, blocks, extents
   b. Platters, cylinders
   c. logical records, blocks, extents
   d. tables and pointers

7. Data formats on the HP 3000 are:

   a. BINARY
   b. EDCDIC
   c. ASCII
   d. a and b
   e. a and c
   f. none of the above

8. Record formats on the HP 3000 are:

   a. fixed, undetermined or variable
   b. fixed length only
   c. variable or fixed
   d. undefined, variable or fixed

9. What is the maximum number of characters in a file name when using a fully qualified file name including a lockword?

10. The Build command can be used to build a file on magnetic tape?

    a. True
    b. False

FILE SYSTEM REVIEW QUESTIONS (CONT'D

11. A file code of –1 would indicate the file must be accessed in privileged mode.

    a. True
    b. False

12. The spooler is a file that has just been spooled.

    a. True
    b. False

13. The STREAM command normally expects the character   in place of the standard MPE ":".

    a. ;
    b. >
    c. !
    d. none of the above

14. A STREAM file created with the EDITOR must be kept unnumbered.

    a. True
    b. False

15. A named USL file created by a compiler will be

    a. temporary
    b. permanent

16. A program file created by the Segmenter (:PREP) is created as a job/session temporary file.

    a. True
    b. False

ANSWERS TO FILE SYSTEMS REVIEW QUESTIONS

1. b, False
2. b, False
3. a, True
4. b, False
5. a, True
6. c
7. e
8. d
9. 35 :filename/lockword.group name.account name including required punctuation.
10. b, False
11. a, True
12. b, False.
13. c
14. a,True
15. b
16. a

LAB 7 (FILES)

Read the entire lab. It will be helpful to write out the commands on paper before doing the lab.

1. Obtain from the instructor a COBOL source deck. Add the appropriate control cards to identify the deck as belonging to your user and account name. Submit the deck to the system to be accessed at a later time in this lab. Issue a :SHOWIN command to confirm the deck is in the system under your user and account name.

2. Compile the COBOL program whose source is in COBL1.PUB; save the results of the compilation in a file named USL.

3. Prepare the results of the compilation above and store the results in a temporary program file called PGM1.

4. Issue the :FILE command that will direct the compiler (invoked from the terminal) to the source deck you submitted in step 1.

5. With one command, compile and prepare the program in step 5. Put the results in a permanent file called PGM2.

6. Change the name of program PGM2 to PROG.

7. Change the name of PGM1 to TEMP.

8. Issue the command to nullify all previously issued FILE commands for your session.

9. Obtain a mag tape from the instructor. Issue the commands to store on tape the file PROG. Use a meaningful name for the tape. Respond at the console to the systems request for I/O.

10. Log-off the system.

11. Log on the system.

12. Issue the commands to load back on to the system, from your tape, <u>only</u> the file PROG. Respond to the console message.

13. Obtain a list of the files under your account.

14. Was TEMP on the system? Why?

Note: If a card reader is not available, the data for this lab can be found in LAB7DATA.PUB. Use the EDITOR to prefix the deck with an appropriate !DATA card. Keep the file unnumbered. STREAM that file to enter the data into the system.

LAB #8 (STREAM)

Log on the system in session mode.

A. Create a batch job stream with the Editor. This job stream should compile, prep, and execute the source file COBSTRM.PUB. Do not use :COBOLGO. Use $OLDPASS and $NEWPASS for your USL and program files where applicable. Remember that using the STREAM command initiates a job independently of the current job or session.

Key Points:

What commands delimit a job?

What character is the STREAM command expecting?

Where will your compilation listing and any program output appear and why?

Are there any special considerations when creating STREAM files with the Editor?

What command can you use to find out the status of your job?

B. Create the same STREAM file in a session <u>without</u> using the Editor.

FCOPY CAPABILITIES

- COPY FILES
- FUNCTION ON FILE SUBSETS
- MANIPULATE MULTIFILE VOLUMES (TAPES)
- PERFORM CODE TRANSLATION
- FILE DUMPS (MULTIPLE FORMATS)
- CREATE NEW FILES
- LOWER/UPPER CHARACTER CONVERSION
- FILE ERROR-HANDLING TECHNIQUES
- PERFORM COPY VERIFICATION
- PERFORM FILE COMPARES

FCOPY

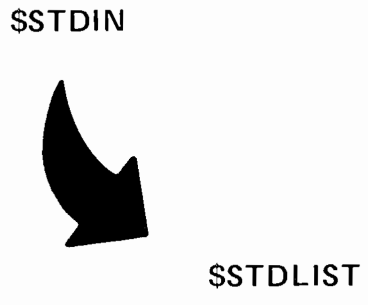
NOTE:
FILE COMMANDS MAY
BE NEEDED

```
:RUN FCOPY.PUB.SYS
```

$$>\text{FROM=}\left\{\begin{matrix}\text{empty}\\ \text{fromfile}\\ *\end{matrix}\right\}\ ;\text{TO=}\ \left\{\begin{matrix}\text{empty}\\ \text{tofile}\\ *\end{matrix}\right\}\ (;\text{OPTIONS})$$

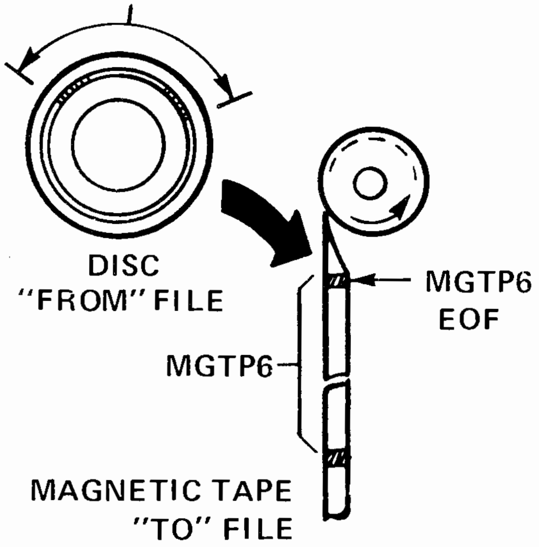
empty

```
:RUN FCOPY.PUB.SYS
>FROM=; TO=
>EXIT
```



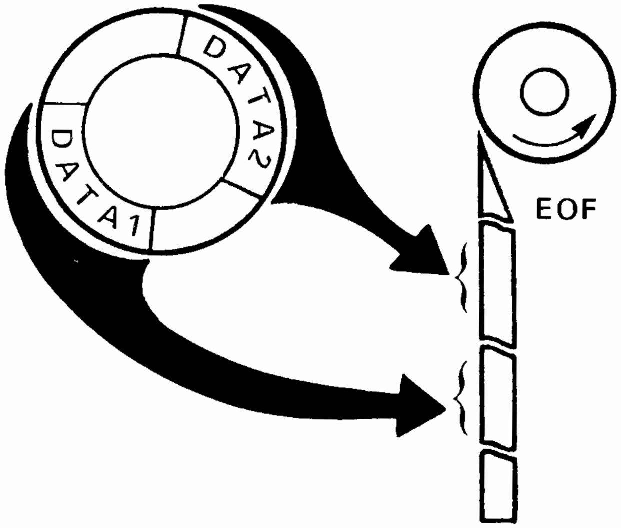
filereference

```
:FILE MGTP6;DEV=TAPE; &
:REC=-80,16,F,ASCII
:RUN FCOPY.PUB.SYS
>FROM=COBLSC;TO=*MGTP6
>EXIT
```



*(internal backreference)

```
:FILE T;DEV=TAPE;REC=40,20
:RUN FCOPY.PUB.SYS
>FROM=DATA1;TO=*T
>FROM=DATA2;TO=*
>EXIT
```



FCOPY (CONT'D)

;NEW ➡ DESTINATION FILE IS A NEW DISC FILE

```
:RUN FCOPY.PUB.SYS

FROM=YOURPGM.YOURGRP.YOURACCT; TO=MYPGM; NEW
```

```
:RUN FCOPY.PUB.SYS

HP32212A.Ø.Ø3  FILE COPIER

>FROM=PCWFILE.PUB.CHONLE;TO=PCWFILE;NEW
EOF FOUND IN FROMFILE AFTER RECORD 44

45 RECORDS PROCESSED *** Ø ERRORS

>EXIT

END OF PROGRAM
```

FCOPY (CONT'D)

;SUBSET	➡	COPY THRU 1ST/NEXT EOF DEFAULT IS ENTIRE TAPE UNTIL FCOPY ENCOUNTERS UNREADABLE DATA
OR		
;SUBSET="char-strg"[,[col.] [,EXCLUDE]	➡	"ANSWER" "YES" " "
OR		
;SUBSET=#patternlist#[,[col.] [,EXCLUDE]	➡	#%276,103,%21,%110,45,16#
OR		
;SUBSET=[startrec] [,numrecs]	➡	501,50
OR		
;SUBSET=[startrec] [:lastrec]	➡	50:100

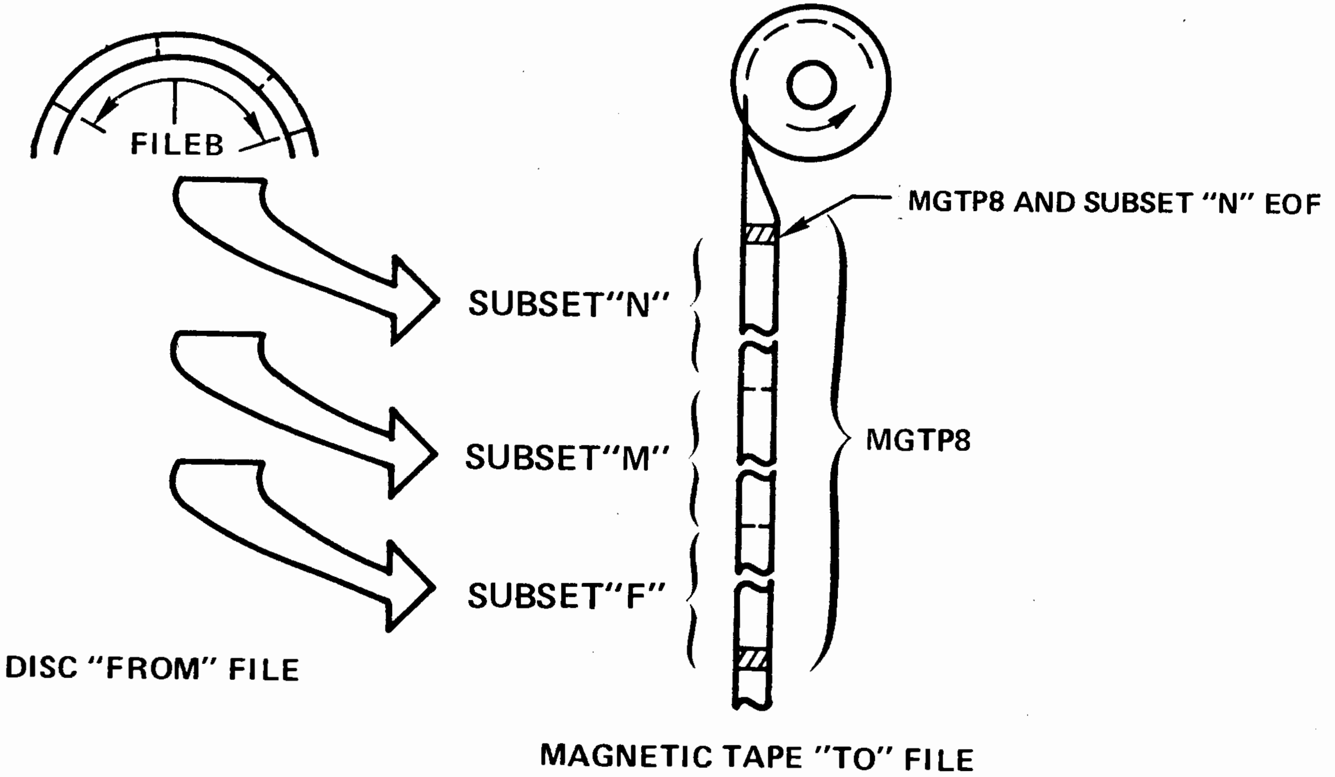
FCOPY (CONT'D)

SUBSET EXAMPLE

```
>FROM=FILEB;TO=*MGTP8;SUBSET="F",2
>FROM=FILEB;TO=*;SUBSET="M",2
>FROM=FILEB;TO=*;SUBSET="N",2
>EXIT
```

- copy records with F in col 2

*ALONE MEANS
CONCATENATION
NO EOF ON
MAG TAPE



FCOPY example

TO DUMP ALL RECORDS FROM "PCWFILE"
THAT HAVE "AMERICAN"
STARTING IN POSITION 7 OF RECORD

```
:RUN FCOPY.PUB.SYS

HP32212A.0.03 FILE COPIER

>FROM=PCWFILE;TO=;SUBSET="AMERICAN",7

WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 72 BYTES.
DO YOU WISH TO CONTINUE THIS OPERATION?  (IF SO, ANSWER YES)

>YES
102303AMERICAN KITCHEN MACHINE CO              76478         48669
102322AMERICAN MARINE IND                             6080
102323AMERICAN METAL RESTAURANT EQUI                 34020
102342AMERICAN PLBG + STM SUPPLY CO   150073 1020132
102363AMERICAN AIR FILTER
103893AMERICAN SUPPLY CO              105186
103911AMERICAN WESTERN SALES INC
EOF FOUND IN FROMFILE AFTER RECORD 44

7 RECORDS PROCESSED *** 0 ERRORS

>EXIT

 END OF PROGRAM
```

FCOPY

TO LIST THE FIRST
10 RECORDS OF "PCWFILE"

```
:FILE PLIST;DEV=LP
:RUN FCOPY.PUB.SYS

HP32212A.0.23 FILE COPIER

>FROM=PCWFILE;TO=*PLIST;SUBSET=0,10

WARNING: FROMFILE RECSIZE IS 80 BYTES, TOFILE RECSIZE IS 132 BYTES.
DO YOU WISH TO CONTINUE THIS OPERATION?  (IF SO, ANSWER YES)

>YES
10 RECORDS PROCESSED *** 0 ERRORS


>EXIT
```

```
          100523ACE HARDWARE                                   124695
          100681ACE FIXTURE CO                         62563
          100723ACME AMERICAN REPAIRS                  15443    47055
          100793ACME EQUIP CO INC
          101132AFFILIATED SUPPLY CO                   4700001015813
          101212AIR REP COMPANY                        29268
          101292ALS SALES AND SERVICE
          101293ALADIN REPAIRS                                  36100
          101442ALAMO PIPE + SUPPLY                    3403101869073
          101563ALAMO PLUMBING SUPPLY
```

FCOPY (CONT'D)

TRANSLATION FUNCTIONS

;EBCDICIN → translate from EBCDIC to ASCII

or

;BCDICIN → translate from BCDIC to ASCII

or

;EBCDICOUT → translate from ASCII to EBCDIC

or

;BCDICOUT → translate from ASCII to BCDIC

;UPSHIFT → DESTINATION FILE CONTENTS WILL BE UPSHIFTED

FCOPY

FUNCTION	PARAMETER
VERIFY TO VERIFY THE ACCURACY OF A COPY IMMEDIATELY AFTER WRITING IT	;VERIFY(=VERRS)
COMPARE TO COMPARE WITHOUT COPYING, THE CONTENTS OF THE "FROM" FILE AND "TO" FILE	;COMPARE(=CERRS)
IGNORE ERRORS (MAG TAPE ONLY) TO IGNORE "READ" ERRORS IN THE FROM FILE. BLOCKS IN ERROR ARE BYPASSED	;IGNERR(=ERRS)

. EACH OF THE ERROR PARAMETERS IS THE NUMBER OF ERRORS, OR COMPARES THAT WILL STOP PROCESSING

1 IS THE DEFAULT

FCOPY (CONT'D)

TO SKIP END-OF-FILE MARKS ON EITHER OR BOTH THE "FROM" FILE AND THE "TO" FILE

;SKIPEOF =n-fromfile,n-tofile → =2,4

or

;SKIPEOF =n-fromfile → =3

or

;SKIPEOF=,n-tofile → =,6

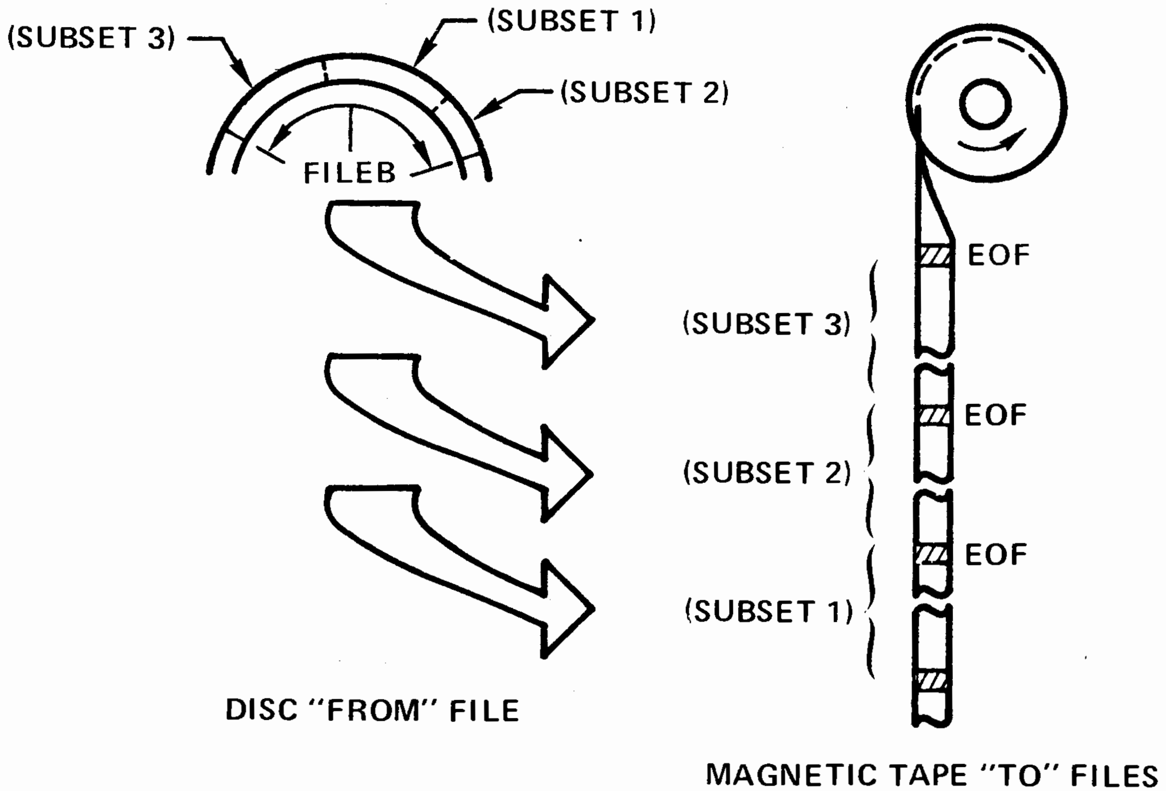
note: MAG TAPES ONLY

to skip files
on magtape

SKIPEOF EXAMPLE

FCOPY (CONT'D)

```
:FILE MGTP4;DEV=TAPE
:RUN FCOPY.PUB.SYS
>FROM=FILEB;TO=*MGTP4;SUBSET=13:45
>FROM=FILEB;TO=*MGTP4;SKIPEOF=,1 ;SUBSET=55:105
>FROM=FILEB;TO=*MGTP4;SKIPEOF=,2 ;SUBSET=:12
>EXIT
```



NOTES ON FCOPY

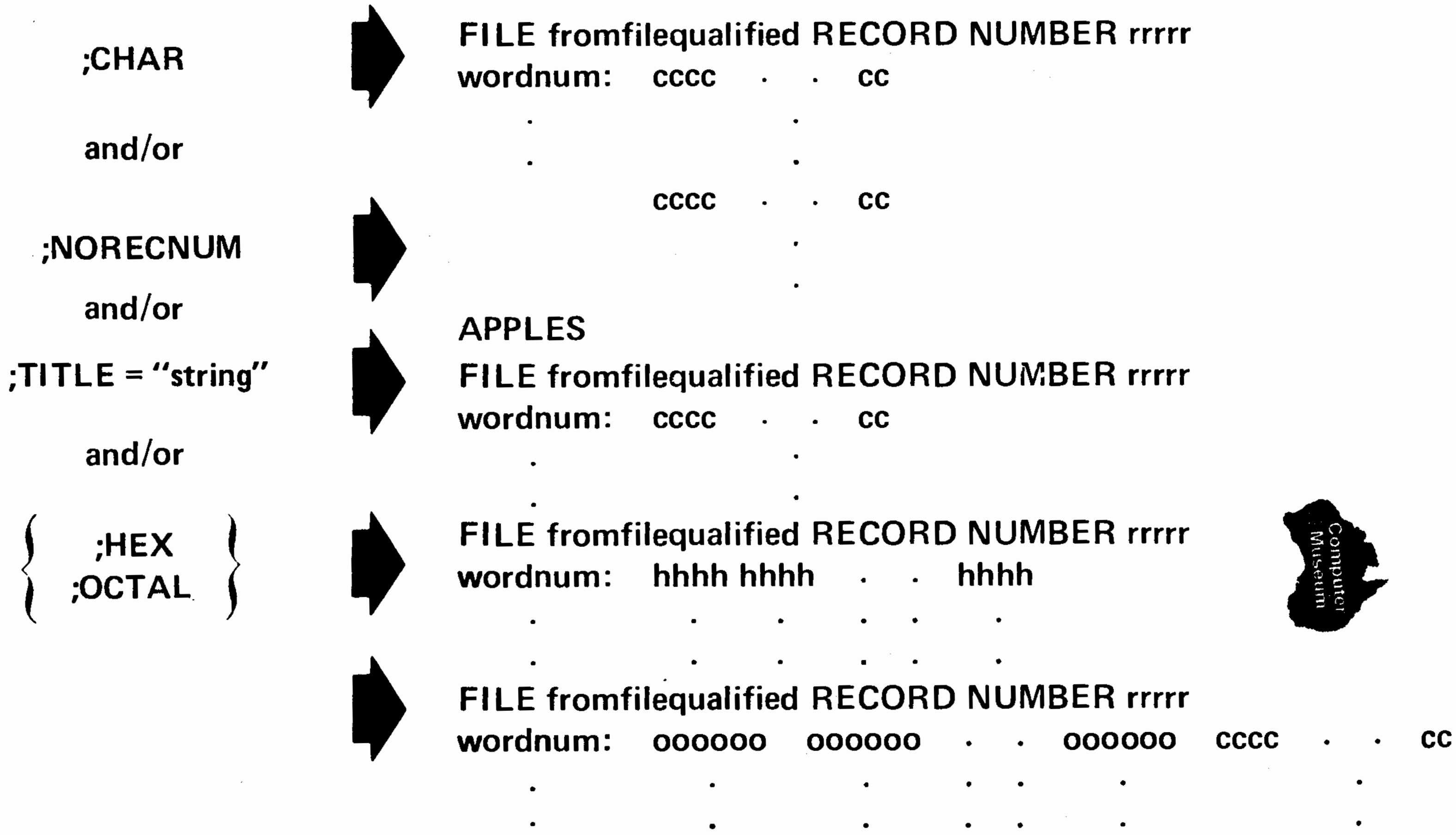
IF A FILE IS REFERENCED IN THE *FILENAME FORMAT, AND THE DEVICE IS A MAGNETIC TAPE OR ANY OTHER PHYSICAL DEVICE WHICH REQUIRES OPERATOR INTERVENTION, THE DEVICE WILL REWIND AND A PROMPT ISSUED ON THE SYSTEM CONSOLE.

FCOPY BACKSPACES OVER THE EOF MARK AFTER THE COPY OPERATION IS COMPLETE. IT IS THEN POSITIONED CORRECTLY IN THE EVENT THAT THE NEXT COPY OPERATION USES THE SAME FILE REFERENCE.

FILES REFERENCED WITH THE INTERNAL REFERENCE (*ONLY) CAUSE THE FILE TO REMAIN POSITIONED IN THE SAME PLACE (AND OPEN) PRIOR TO THE NEXT COPY OPERATION.

FCOPY (CONT'D)

DUMP FORMATS



;CHAR

and/or

;NORECNUM

and/or

;TITLE = "string"

and/or

;HEX
;OCTAL

```
FILE fromfilequalified RECORD NUMBER rrrrr
wordnum:   cccc  .  .  cc
     .              .
     .              .
           cccc  .  .  cc
                    .
                    .
```

```
APPLES
FILE fromfilequalified RECORD NUMBER rrrrr
wordnum:   cccc  .  .  cc
     .              .
     .              .
```

```
FILE fromfilequalified RECORD NUMBER rrrrr
wordnum:   hhhh hhhh  .  .  hhhh
     .        .    .  .  .   .
     .        .    .  .  .   .
```

```
FILE fromfilequalified RECORD NUMBER rrrrr
wordnum:   oooooo   oooooo  .  .  oooooo   cccc  .  .  cc
     .        .        .    .  .    .            .
     .        .        .    .  .    .            .
```

FCOPY

```
:FILE PLIST;DEV=LP
:RUN FCOPY.PUB.SYS

>FROM=PCWFILE;TO=*PLIST;SUBSET=Ø,1Ø;CHAR;OCTAL;TITLE="CHAR/OCTAL"
>EXIT



CHAR/OCTAL

FILE PCWFILE.PUB.COMUTLZ RECORD NUMBER 0
000000:  030460 030005 031061 040503 042440 044101 051104 053501 051105 020040 020040 020040 100521ACE HARDWARE
000014:  020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 030462                         12
000030:  032060 034465 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 4695
000044:  020040 020040 020040 020040

FILE PCWFILE.PUB.COMUTLZ RECORD NUMBER 1
000000:  030460 030066 030061 040503 042440 043111 054124 052522 042440 041517 020040 020040 100601ACE FIXTURE CO
000014:  020040 020040 020040 020040 020040 020040 020040 020065 031065 034063 020040 020040                    62583
000030:  020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
000044:  020040 020040 020040 020040

FILE PCWFILE.PUB.COMUTLZ RECORD NUMBER 2
000000:  030460 030067 031063 040503 046505 020101 046505 051111 041501 047040 051105 050101 100723ACME AMERICAN REPA
000014:  044522 051440 020040 020040 020040 020040 020040 020061 032464 032063 020040 020064 IRS                 15443    4
000030:  033460 032465 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 7055
000044:  020040 020040 020040 020040

FILE PCWFILE.PUB.COMUTLZ RECORD NUMBER 3
000000:  030460 030067 034463 040503 046505 020105 050525 044520 020103 047440 044516 041440 100793ACME EQUIP CO INC
000014:  020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
000030:  020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
000044:  020040 020040 020040 020040

FILE PCWFILE.PUB.COMUTLZ RECORD NUMBER 4
000000:  030460 030461 031462 040506 043111 046111 040524 042504 020123 052520 050114 054440 101132AFFILIATED SUPPLY
000014:  041517 020040 020040 020040 020040 020040 020040 020064 033460 030060 030061 030061 CO                  470000101
000030:  032470 030463 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 5813
000044:  020040 020040 020040 020040

   •
   •
   •
```

FCOPY

```
:FILE PLIST;DEV=LP
:RUN FCOPY.PUB.SYS

>FROM=PCWFILE;TO=*PLIST;SUBSET=Ø,10;CHAR;HEX;NORECNUM
>EXIT

     3130 3035 3231 4143 4520 4841 5244 5741 5245 2020 2020 2020 2020 2020 2020 2020 100521ACE HARDWARE
     2020 2020 2020 2020 2020 2020 2020 3132 3436 3935 2020 2020 2020 2020 2020 2020                124695
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3036 3031 4143 4520 4649 5854 5552 4520 434F 2020 2020 2020 2020 2020 2020 100601ACE FIXTURE CO
     2020 2020 2020 2036 3235 3833 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020        62583
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3037 3233 4143 4D45 2041 4D45 5249 4341 4E20 5245 5041 4952 5320 2020 2020 100723ACME AMERICAN REPAIRS
     2020 2020 2020 2031 3534 3433 2020 2034 3730 3535 2020 2020 2020 2020 2020 2020        15443   47055
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3037 3933 4143 4D45 2045 5155 4950 2043 4F20 494E 4320 2020 2020 2020 2020 100793ACME EQUIP CO INC
     2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3131 3332 4146 4649 4C49 4154 4544 2053 5550 504C 5920 434F 2020 2020 2020 101132AFFILIATED SUPPLY CO
     2020 2020 2020 2034 3730 3030 3031 3031 3538 3133 2020 2020 2020 2020 2020 2020        4700001015813
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3132 3132 4149 5220 5245 5020 434F 4D50 414E 5920 2020 2020 2020 2020 2020 101212AIR REP COMPANY
     2020 2020 2020 2032 3032 3638 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020        20268
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3132 3932 414C 5320 5341 4C45 5320 414E 4420 5345 5256 4943 4520 2020 2020 101292ALS SALES AND SERVICE
     2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3132 3933 414C 4144 494E 2052 4550 4149 5253 2020 2020 2020 2020 2020 2020 101293ALADIN REPAIRS
     2020 2020 2020 2020 2020 2020 2020 2033 3631 3030 2020 2020 2020 2020 2020 2020                36100
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3134 3432 414C 414D 4F20 5049 5045 202B 2053 5550 504C 5920 2020 2020 2020 101442ALAMO PIPE + SUPPLY
     2020 2020 2020 2033 3430 3331 3031 3836 3930 3733 2020 2020 2020 2020 2020 2020        3403101869073
     2020 2020 2020 2020 2020 2020 2020 2020

     3130 3135 3033 414C 414D 4F20 504C 554D 4249 4E47 2053 5550 504C 5920 2020 2020 101503ALAMO PLUMBING SUPPLY
     2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 2020
     2020 2020 2020 2020 2020 2020 2020 2020
```

FCOPY and HP 2644A
(sessions only)

$CTUL	CARTRIDGE TAPE UNIT (LEFT)
$CTUR	CARTRIDGE TAPE UNIT (RIGHT)
$HARD	13246A/B THERMAL LINE PRINTER (LOCAL TO 2644 ONLY)

* CANNOT BE USED TO SPECIFY A PRIOR USE OF $CTUL, $CTUR, OR $HARD

TO COPY FROM 2 DISC FILES TO RIGHT TAPE UNIT AND SEPARATE EACH FILE ON THE TAPE BY AN END-OF-FILE MARK.

```
:RUN FCOPY.PUB.SYS
>FROM=DISCFL1;TO=$CTUR
>FROM=DISCFL2;TO=$CTUR;SKIPEOF=,1
>EXIT
```

TO COPY THE THIRD FILE FROM LEFT TAPE UNIT TO THE TERMINAL SCREEN ($STDLIST).

```
:RUN FCOPY.PUB.SYS
>FROM=$CTUL;TO=;SKIPEOF=2
>EXIT
```

FCOPY AND HP 2644A

TERMINAL SETTINGS

- HALF DUPLEX
- ECHO OFF
- AUTOMATIC CARRIAGE RETURN/LINE FEED OFF
- PARITY OFF

NOTE:

FCOPY HAS NO WAY TO DETERMINE WHETHER OR NOT THE USER IS ACTUALLY USING AN HP2644A TERMINAL. CONSEQUENTLY, THE EFFECTS OF USING THESE FEATURES OF FCOPY WITH ANY OTHER TYPE OF TERMINAL ARE UNPREDICTABLE.

FCOPY

TO TERMINATE FCOPY

```
> EXIT
```

- INPUT AND OUTPUT FILES WILL BE CLOSED
- MAGNETIC TAPES REWIND AND GO OFF LINE
- SPOOLED LINE PRINTERS DO NOT START OUTPUT UNTIL FILE IS CLOSED

FCOPY EXERCISES

(FILL IN NECESSARY PARAMETERS)

FROM A SESSION:

TAPE TO TAPE, 80 BYTE ASCII RECORDS, INPUT TAPE BLOCKED 1, OUTPUT TAPE BLOCKED 20.

```
:HELLO
:FILE
:FILE
:RUN FCOPY.PUB.SYS
>FROM=    ;TO=
>EXIT
:BYE
```

FROM A JOB:

COPY FROM A DISC FILE NAMED INDISC, TO THE LINE PRINTER, ONLY THOSE RECORDS WITH "2" IN POSITION 20 OF THE INPUT RECORD.

```
:JOB
:RUN FCOPY.PUB.SYS
FROM=    ;TO=
EXIT
:EOJ
```

FCOPY WORK EXERCISE

Write out the commands to complete the tasks listed below.

A tape has 3 separate data files, each separated by an EOF. Using the file copier (FCOPY) put each file in a different disc file.

The files on tape have 28,50, and 27 records respectively. Each file has a record format of 80 bytes, blocked 16, fixed, ASCII. Try to structure your commands so that you get only one prompt at the console.

Keep in mind where the tape is positioned at the end of each operation and use the SKIPEOF option.

You also have the option to BUILD the files or use the NEW option on FCOPY. When would you want to do a BUILD and when would you want the NEW option?

Write the commands to copy the three tape files above into 1 disc file.

SOLUTIONS TO FCOPY WORK EXERCISE

To copy three files off one tape to three disc files:

```
:FILE TAPEIN;DEV=TAPE;REC=-80,16,F,ASCII
:BUILD DISCFL1;REC=-80,16,F,ASCII;DISC=30
:BUILD DISCFL2;REC=-80,16,F,ASCII;DISC=60
:BUILD DISCFL3;REC=-80,16,F,ASCII;DISC=30
:RUN FCOPY.PUB.SYS
>FROM=*TAPEIN;TO=DISCFL1;SUBSET
>FROM=*;TO=DISCFL2;SUBSET;SKIPEOF=1
>FROM=*;TO=DISCFL3;SUBSET;SKIPEOF=1
>EXIT
```

To copy three files off of one tape into one disc file:

```
:FILE TAPEIN;DEV=TAPE;REC=-80,16,F,ASCII
:RUN FCOPY.PUB.SYS
>FROM=*TAPEIN;TO=DISCFILE;NEW:SUBSET
>FROM=*;TO=*;SUBSET;SKIPEOF=1
>FROM=*;TO=*;SUBSET;SKIPEOF=1
>EXIT
```

SORT / MERGE CAPABILITIES

- SORT ANY FILE
- MERGE ANY SORTED FILES
- SORT INTO ASCENDING/DESCENDING ORDER
- CONTIGUOUS, SEPARATED OR OVERLAPPED KEYS
- MULTIPLE DATA TYPE KEYS
- FIXED OR VARIABLE LENGTH RECORDS
- MULTIPLE I/O MEDIA TYPES
- OUTPUT RECORDS, KEYS, REC#'S OR KEYS+REC#'S
- PROGRAMMATIC ACCESS FROM SPL, FORTRAN & COBOL

SORT/MERGE

FILE COMMANDS MAY BE NEEDED
:RUN SORT.PUB.SYS

$$\textbf{INPUT} \quad \left\{\begin{matrix}\text{filename}\\ *\end{matrix}\right\} \left[\text{,number of records}\right] \left[\text{,recsize}\right]$$

filename	➡	any formal designator
*	➡	RECORDS INPUT TO SORT FOLLOW THE SORT END COMMAND FROM $STDINX
num-of-recs	➡	disc current EOF; non-disc default 10,000
recsize	➡	ignored unless * SPECIFIED THEN MAY BE ≤$STDINX FILE RECORD SIZE

SORT/MERGE (CONT'D)

OUTPUT $\left\{\begin{matrix}\text{filename}\\ *\end{matrix}\right\}$ [,NUM] [,KEY]

filename	→	any formaldesignator
*	→	OUTPUT RECORDS TO $STDLIST ONLY
NUM	→	double integer record numbers only
KEY	→	keys concatenated Left to Right MAJOR FIRST
NUM & KEY	→	double integer record number followed by concatenated keys
NEITHER NUM NOR KEY		output records same as input records in sorted order

SORT/MERGE (CONT'D)

KEY position [,length] [,type] [,DESC] [;position . . .]

length ▶ number = 1ST column of KEY position

type ▶ number = total KEY Length (bytes)

NOTE:
length parameter required
for BYTE, PACKED, PACKED*
and DISPLAY data formats.

BYTE > ASCII, EBCDIC, or LOGICAL (default)

INT > 2's complement integer – default = 2 bytes

DOUBLE > 2's complement integer – default = 4 bytes

REAL > HP 3000 real format – default = 4 bytes

LONG > HP 3000 long format – default = 8 bytes

PACKED >

d	d	d	d	d	d	d	S

PACKED* > SEE SORT/MERGE MANUAL

DISPLAY >

+0 ↘ 9 = (↘ I	–0 ↘ 9 =) ↘ R	0 ↘ 9 = 60 ↘ 71

DESC ▶ descending for this key – default = ascending

SORT /MERGE (CONT'D)

RESET → START KEY SPECIFICATIONS OVER

VERIFY → LIST OPTIONS IN EFFECT

INPUT FILE = FILEA

NUMBER OF RECORDS = 10,000

OUTPUT FILE = *,KEY

KEY POSITION	LENGTH	TYPE	ASC/DESC	
50	5	BYTE	ASC	(MAJOR KEY)
1	10	DISPLAY	DESC	
21	8	PACKED	ASC	

END → START SORT OR PROMPT WITH "?" IF "*" SPECIFIED AS INPUT

:EOD → START SORT (end of data input)

SORT/MERGE (CONT'D)

CONTROL Y (DURING EXECUTION – INTERACTIVE ONLY)

```
INPUT PHASE:   1234 RECORDS HAVE BEEN INPUT
INTERMEDIATE SORT PHASE:  PASS 1 OF 3
OUTPUT PHASE:   1234 RECORDS HAVE BEEN OUTPUT
```

SORT COMPLETION

SAMPLE STATISTICS

NUMBER OF RECORDS =	100,000
RECORD SIZE (IN BYTES) =	100
NUMBER OF INTERMEDIATE PASSES =	2
SPACE AVAILABLE (IN WORDS) =	18,521
NUMBER OF COMPARES =	1,734,091
NUMBER OF SCRATCHFILE IO'S =	9,921
CPU TIME (MINUTES) =	44.19
ELAPSED TIME (MINUTES) =	63.95

SORT/ MERGE (CONT'D)

FILE COMMANDS MAY BE NEEDED
:RUN MERGE.PUB.SYS

INPUT filename [,filename]. . . ALL IN ONE COMMAND

INPUT FILE MUST BE IN SORTED ORDER

OUTPUT {Filename / $STDLIST} [,numrecs] [,KEY]

filename	➡	formalfiledesig (default – "OUTPUT")
$STDLIST	➡	Listed to output device (file not saved)
numrecs	➡	only if one or more non-disc files
KEY	➡	Only keys in OUTPUT

RESET
KEY } SAME AS SORT

SORT/MERGE (CONT'D)

VERIFY LISTS OPTIONS IN EFFECT

```
VERIFY
    INPUT FILES = MASTER, UPDATE
    OUTPUT FILE = NEWMASTER
    KEY POSITION     LENGTH     TYPE      ASC/DESC
          50             5      BYTE        ASC       (MAJOR KEY
           1            10      DISPLAY     DESC
          21             8      PACKED      ASC
```

END START MERGE

CONTROL Y number RECORDS HAVE BEEN OUTPUT

```
MERGE COMPLETION
 SAMPLE     Statistics

                NUMBER OF INPUT FILES =                   3
                NUMBER OF RECORDS =                 100,000
                RECORD SIZE (IN BYTES) =                100
                SPACE AVAILABLE (IN WORDS) =         15,325
                NUMBER OF COMPARES =                167,012
                CPU TIME (MINUTES) =                   3.25
                ELAPSED TIME (MINUTES) =               9.73
```

SORT EXAMPLE (session)

TAPE FILE INPUT; DISC FILE OUTPUT
SORT ON 2 ASCII FIELDS, MAJOR KEY IN ASCENDING ORDER,
MINOR KEY IN DESCENDING ORDER

```
:HELLO USERNAME.ACCTNAME
:FILE INPUT=TPIN;DEV=TAPE ;REC=-30,51,F,ASCII
:RUN SORT.PUB.SYS
>OUTPUT DISCFL
>KEY 1,10;20,2;DESC
>END

    STATISTICS
    .
    .
    .

:BYE
```

SORT EXAMPLE (JOB)

DISC FILE INPUT, DISC FILE OUTPUT
SORT ON 1 FIELD OF REAL DATA STARTING IN POSITION 5, ASCENDING

```
:JOB USERNAME.ACCTNAME
:RUN SORT.PUB.SYS
INPUT DISCFL1
OUTPUT DISCFL2
KEY 5, REAL                (LENGTH OF 4 ASSUMED BY DATA TYPE)
END
:EOJ
```

STATISTICS WILL APPEAR ON $STDLIST

LISTCRET CAPABILITIES

- LISTS CREATOR OF STORE/SYSDUMP TAPES
  - HANDLES MULTIPLE TAPE REELS
    - LISTS FULLY QUALIFIED FILE NAMES
      - LISTING CAN BE REDIRECTED (FTN06)

```
:RUN LISTCRET.PUB.SYS

IS THE TAPE A CONTINUATION REEL (Y/N) ? N

STORE TAPE LIST PROGRAM*********************

FILE NAME                          CREATOR

DBMSLABS.PUB     .JOHNSON          BOB
DEEPSC    .PUB     .JOHNSON          BOB
UTLZ      .PUB     .JOHNSON          BOB
  END OF PROGRAM
```

NOTE:
OPERATOR WILL BE
PROMPTED FOR "FTN01"

➡ ALLOWS DISCOVERY OF ATTRIBUTES
OF ANY UNKNOWN TAPE

TAPECOPY CAPABILITIES

COPIES/VERIFIES MULTIPLE COPIES

- STORE TAPE
  - SINGLE FILE
    - COLDLOAD TAPE

NOTE: OPERATOR WILL BE PROMPTED FOR "MASTER" AND "COPY"

```
:RUN TAPECOPY.PUB.SYS

HP 3ØØØ TAPE COPY AND VERIFY PROGRAM
VERSION 2.Ø
ENTER FORMAT (STORE/COLD LOAD/SINGLE FILE)
STORE
BEGIN COPYING . . . COPIES=1
39 RECORDS COPIED
BEGIN VERIFICATION
COPY1 IS OK.
MORE COPIES?
NO

END OF PROGRAM
```

➡ TAPE DENSITY CHANGES

➡ QUICK & EFFICIENT

KONVERT

- CONTRIBUTED PROGRAM
- CAN SPECIFY WHERE PACKED DECIMAL FIELDS ARE IN RECORDS.
- WILL ALSO HANDLE ODD BYTE LENGTH BLOCKS
- EBCDIC TO ASCII
- RECORD SIZE = OR < 4096 BYTES
- UP TO 1000 PACKED DECIMAL FIELDS PER RECORD

EXAMPLE:

```
 :FILE TPIN;REC=-80,20,F,ASCII;DEV=TAPE
 :FILE DSKOUT,NEW;REC=-80,16,F,ASCII;SAVE
 :RUN KONVERT.PUB.SYS
> FROM=*TPIN;TO=*DSKOUT;SKIPEOF=1
> 20,4
> 60,6
> CR
```

SORT/FCOPY EXERCISE

TASK:

Write out the commands to run the SORT from your session using card input and tape output. List the sorted file on the line printer.

NOTES:

1. Tape output of the SORT should be reblocked to 16 records per block.
2. Sort fields:

   Major= position 2, length 2

   Minor= position 45, length 34
3. When using the file copier
   a. In the related FILE commands, make sure the "from" and "to" record sizes are the same to supress the warning message.
   b. Include the parameter that will prevent the file copier from ending the copy operation on an error.

TASK:

Write out and submit to the instructor, the commands to run the same task above, from a job. The SORT input data is to be submitted in-line with the job control cards.

solutions to SORT/FCOPY exercise

```
          :DATA user.account
             .
             .
          :EOD

(session) :HELLO user.account
          :FILE CDIN;DEV=CARD
          :FILE TPOUT;DEV=TAPE;REC=-80,16,F,ASCII
          :RUN SORT.PUB.SYS
           INPUT *CDIN
           OUTPUT *TPOUT
           KEY 2,2;45,34
           END
          :FILE PRINT;DEV=LP;REC=-80,16,F,ASCII
          :RUN FCOPY.PUB.SYS
          >FROM=*TPOUT;TO=*PRINT;SUBSET
          >EXIT

          :BYE


(JOB)

          :JOB user.account
          :FILE TPOUT;DEV=TAPE;REC=-80,16,F,ASCII
          :RUN SORT.PUB.SYS
           INPUT *,200,80
           OUTPUT *TPOUT
           KEY 2,2;45,34
           END
             .         }
             .         }  user input data
             .         }
          :EOD
          :RUN FCOPY.PUB.SYS
          >FROM=*TPOUT;TO=
          >EXIT
```

LAB #9 (UTILITIES)

Read the lab and write out the commands on paper before you start the Lab.

1. Log on the system.

2. A sorted employee record file (EMPDATA.PUB) has this record format:

1-5	6-25	26-27	28	29-30	31	32-34	35-80
EMPL. NO	NAME	AGE	SEX (1-M, 2=F)	YEARS SERVICE	X	JOB CODE	▨

3. Obtain from the instructor additional employee records (unsorted). Add the appropriate control cards to the deck to identify them as belonging to your user and account name. Submit the data to the system.

4. Build a permanent disc file named DFILE. (ASCII, fixed, blocked 16)

5. Using FCOPY and any necessary FILE commands, read the deck you submitted in step 3 into the file you created in step 4.

6. Sort DFILE by years of service (longest first) and put the output back in the same file.

7. Merge this new sorted file (DFILE) with the existing sorted employee file (EMPDATA.PUB) and put the output in another permanent file named MFILE.

8. Using FCOPY and any necessary file commands, copy MFILE to the line printer deleting the fields from job code to the end of the record.

FEATURES OF IMAGE/3000

- ★ PROVIDES POWERFUL SOFTWARE TOOLS TO DEFINE AND CREATE A DATA BASE
- ★ NETWORK DATA STRUCTURE ALLOWING CROSS-REFERENCED ACCESS TO COLLECTIONS OF DATA
- ★ DATA SETS AND INTERRELATIONSHIPS DEFINED ONLY ONCE
- ★ REDUCES DATA REDUNDANCY
- ★ APPLICATION PROGRAMMERS NEED NOT BE CONCERNED ABOUT DETAILS OF ACCESSING THE DATA BASE
- ★ HOST LANGUAGES CAN BE COBOL, FORTRAN, SPL, OR RPG
- ★ SPONTANEOUS AND UNANTICIPATED INQUIRY TO THE EXTERNAL USER THROUGH QUERY/3000
- ★ FLEXIBLE SECURITY SYSTEM AT THE DATA BASE, DATA SET, AND DATA ITEM LEVELS

IMAGE *TERMINOLOGY*

- DATA ITEM – Smallest accessible element of information (FIELD)
- DATA ENTRY – An ordered collection of related data items (RECORD)
- DATA SET – A collection of data entries sharing a common definition
- DATA BASE – A named collection of data sets which are installation owned and fulfill the requirements of all applications which access it and which are structured to model the natural data relationships that exist in a company.
- DATA BASE MANAGEMENT SYSTEM – A tool which enables the user to build a framework of data which, when properly related, can generate meaningful information

IMAGE/3000

SPECIFICATIONS

DATA ITEM NAMES PER DATA BASE: 255

DATA ITEM NAMES PER DATA ENTRY: 127

DATA SETS PER DATA BASE: 99

DETAIL DATA SETS PER MASTER DATA SET: 16

SEARCH ITEMS (KEYS) PER DETAIL DATA SET: 16

MAXIMUM ENTRY SIZE: 4094 BYTES

ENTRIES PER DATA SET: $2^{23} - 1$ (8,388,608)

ENTRIES PER CHAIN: 65,000

CHARACTERS PER DATA BASE NAME: 6

CHARACTERS PER LEVEL WORD NAME: 8

CHARACTERS PER DATA SET NAME: 16

CHARACTERS PER DATA ITEM NAME: 16

IMAGE SUBSYSTEMS

○ DATA BASE DEFINITION SUBSYSTEM (DBDS)

- Used to define all aspects of the data base (SCHEMA)
- Defines data items, security levels, and relationships between data sets

○ DATA BASE MANAGEMENT SUBSYSTEM (DBMS)

- Provides the means for application programmers to access an image data base
- Set of stored library routines invoked by call statements in host-language application programs

○ DATA BASE UTILITY SUBSYSTEM (DBUS)

- Stand-alone utility programs used for creating and maintaining data bases
- Used to create, erase, purge, store, restore, load, and unload data bases
- Assists in restructuring data bases

IMAGE ACCESS METHODS

- SERIAL
- DIRECTED
- CALCULATED (MASTERS ONLY)
- CHAINED

MASTER DATA SETS

- SERVE AS INDEXES TO RELATED DETAIL DATA SET CHAINS
- CONTAIN ONE SEARCH ITEM AND UNIQUE SEARCH ITEM VALUES
- MAY BE RELATED TO UP TO 16 DETAIL DATA SETS
- RELATIVE RECORD LOCATION ASSIGNED TO EACH MASTER ENTRY IS DETERMINED BY PASSING ITS ASSOCIATED SEARCH ITEM OR KEY VALUE THROUGH AN ADDRESS CALCULATION ALGORITHM
- TWO TYPES OF MASTER DATA SETS

  1. MANUAL
  2. AUTOMATIC

DATA BASE MANAGEMENT SUBSYSTEM (DBMS)

DEFINITION: A SET OF STORED LIBRARY ROUTINES INVOKED BY CALL STATEMENTS IN HOST LANGUAGE APPLICATION PROGRAMS.

FUNCTIONS: A MEANS FOR APPLICATION PROGRAMMERS TO ACCESS AN IMAGE DATA BASE.

- SERVES AS THE INTERFACE BETWEEN THE DATA BASE AND THE APPLICATION PROGRAMS
- INITIATES USER ACCESS (OPENING A DATA BASE)
- READS AND UPDATES DATA ITEMS
- READS, WRITES AND DELETES DATA ENTRIES
- RETURNS NAME, STRUCTURE, AND ORGANIZATION INFORMATION
- TERMINATES USER ACCESS (CLOSING A DATA BASE)

ACCESSING DATA BASES (INTRINSICS)

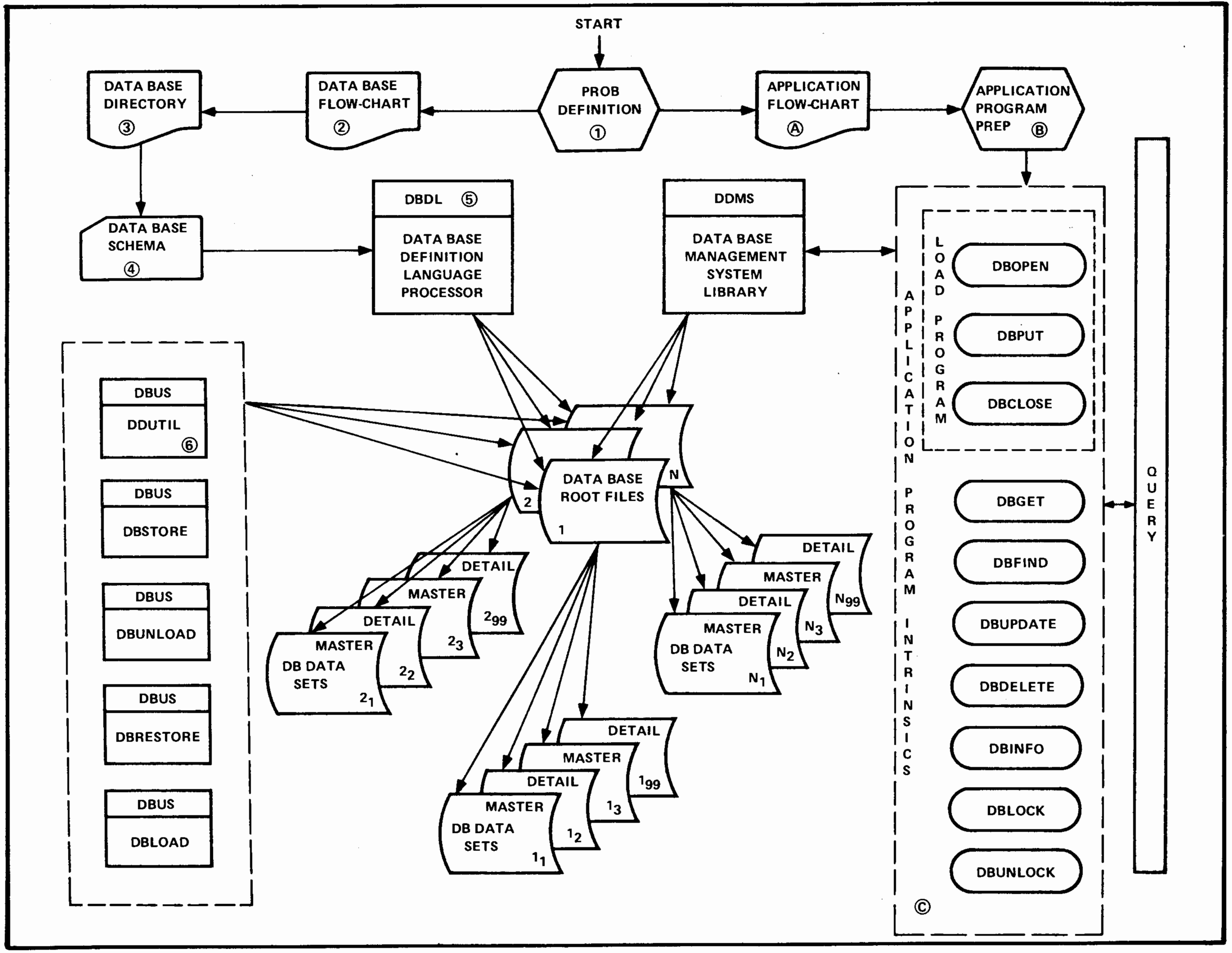
DBOPEN	(BASE, PASSWORD, MODE, STATUS)
DBLOCK	(BASE, DSET, MODE, STATUS)
DBFIND	(BASE, DSET, MODE, STATUS, ITEM, ARG)
DBGET	(BASE, DSET, MODE, STATUS, LIST, BUFFER, ARG)
DBUPDATE	(BASE, DSET, MODE, STATUS, LIST, BUFFER)
DBPUT	(BASE, DSET, MODE, STATUS, LIST, BUFFER)
DBDELETE	(BASE, DSET, MODE, STATUS)
DBINFO	(BASE, QUALIFIER, MODE, STATUS, BUFFER)
DBUNLOCK	(BASE, DSET, MODE, STATUS)
DBCLOSE	(BASE, DSET, MODE, STATUS)

QUERY/3000
FEATURES AND ADVANTAGES

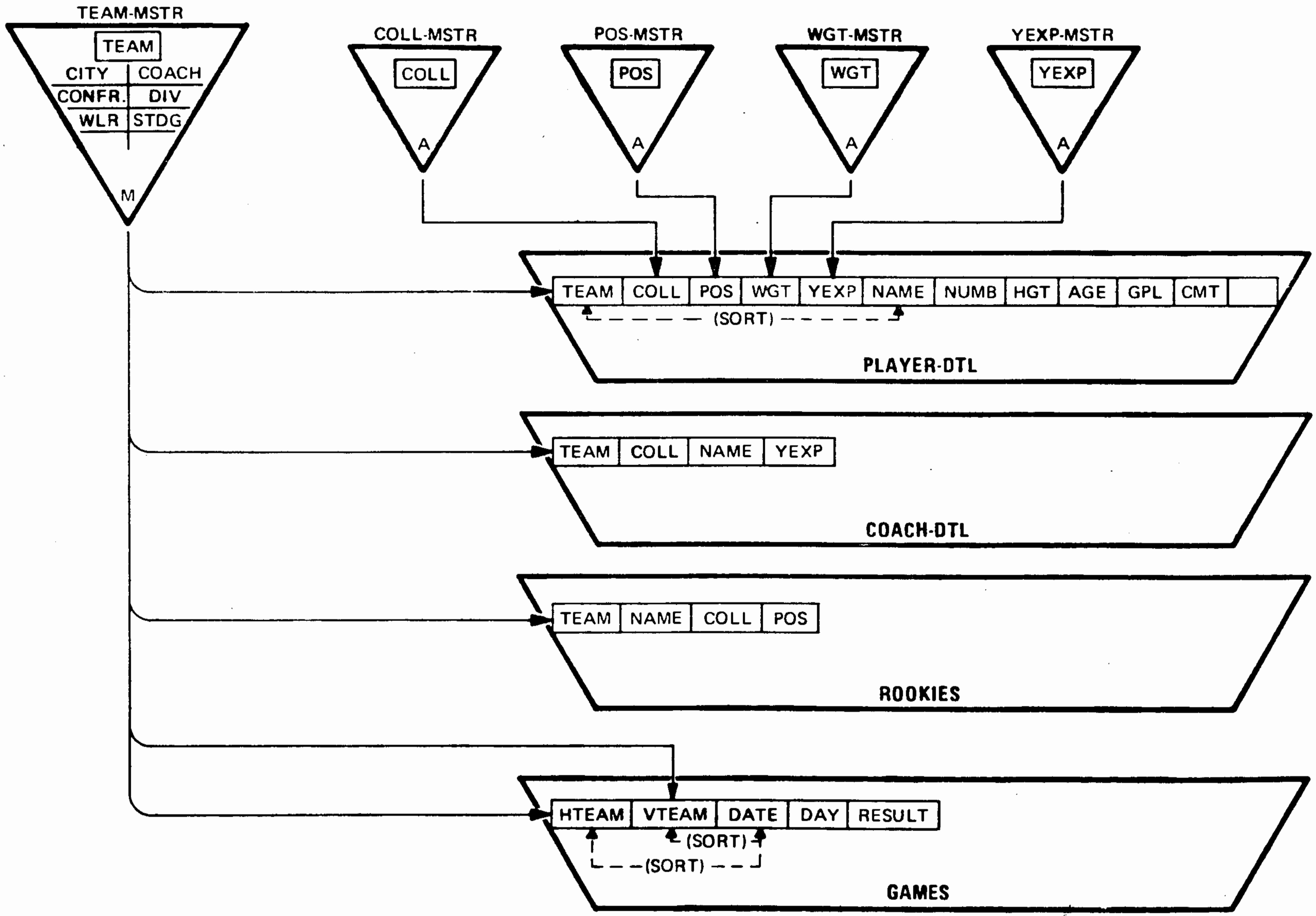
- PROVIDES A SIMPLE METHOD OF DATA BASE ACCESS WITHOUT PROGRAMMING EFFORT
- SELF-CONTAINED SUBSYSTEM INTERFACING WITH DBMS
- ADHERES TO IMAGE/3000 SECURITY PROVISIONS
- TERMINAL/BATCH CAPABILITY
- SELECTS DATA THROUGH LOGICAL COMPARISONS (FIND COMMAND)
- PERMITS SIMPLE DATA:
  - RETRIEVAL
  - REPORTING (FORMATTED OR UNFORMATTED)
  - UPDATING
  - ADDITION
  - DELETION
- MAY BE USED TO CREATE AND STORE QUERY PROCEDURES IN A PROC-FILE
- MAY BE USED TO DISPLAY THE DATA BASE STRUCTURE

QUERY/3000 APPLICATIONS

- CASUAL INQUIRY OF THE DATA BASE
- DATA BASE MODIFICATION (LOW VOLUME)
  - DATA ENTRY ADDITION/DELETION
  - DATA ITEM VALUE MODIFICATION
- REPORT GENERATION
- APPLICATION PROGRAM DEBUGGING


START
PROB DEFINITION ①
DATA BASE FLOW-CHART ②
DATA BASE DIRECTORY ③
DATA BASE SCHEMA ④
APPLICATION FLOW-CHART Ⓐ
APPLICATION PROGRAM PREP Ⓑ
DBDL ⑤
DATA BASE DEFINITION LANGUAGE PROCESSOR
DDMS
DATA BASE MANAGEMENT SYSTEM LIBRARY
DBUS
DDUTIL ⑥
DBUS
DBSTORE
DBUS
DBUNLOAD
DBUS
DBRESTORE
DBUS
DBLOAD
DATA BASE ROOT FILES
1
2
N
DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
2_1
2_2
2_3
2_{99}
DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
1_1
1_2
1_3
1_{99}
DETAIL
MASTER
DETAIL
MASTER
DB DATA SETS
N_1
N_2
N_3
N_{99}
APPLICATION PROGRAM INTRINSICS
LOAD PROGRAM
DBOPEN
DBPUT
DBCLOSE
DBGET
DBFIND
DBUPDATE
DBDELETE
DBINFO
DBLOCK
DBUNLOCK
Ⓒ
QUERY

IMAGE/3000 NFL DATA BASE



HP/3000 COMPILER COMMANDS

$ [$] commandname [parameterlist]

- TITLE COMMAND

$ [$] TITLE [string[,string] . . .]

- Positions 29–132
- string = "|←— 104 —→|"

```
$ TITLE "THIS PROGRAM HAS" ,&
$ "A TITLE."
```

→ THIS PROGRAM HAS A TITLE

HP/3000 COMPILER COMMANDS

- PAGE TITLE & EJECTION

  $ [$] PAGE [string [,string]

  ▶ string = same as for the TITLE command

- CONDITIONAL COMPILATIONS

  $\$ \; [\, \$ \,] \; \text{IF} \; [X_n = \left\{\begin{matrix}\text{OFF}\\ \text{ON}\end{matrix}\right\}] \qquad \$ \; [\, \$ \,] \; \text{SET} \; [X_n = \left\{\begin{matrix}\text{OFF}\\ \text{ON}\end{matrix}\right\} \; [,X_n = \left\{\begin{matrix}\text{OFF}\\ \text{ON}\end{matrix}\right\} \;] \ldots]$

  X_n ▶ XØ ⟶ X9 SWITCHES

  $\left\{\begin{matrix}\text{OFF}\\ \text{ON}\end{matrix}\right\}$ ▶ STATE OF SWITCH

  IF STATE "TRUE", THEN COMPILE

HP/3000 COMPILER COMMANDS

- $ [$] CONTROL parameterlist
  - LIST/NOLIST
  - SOURCE/NOSOURCE
  - WARN/NOWARN
  - MAP/NOMAP
  - CODE/NOCODE
  - $\left\{\begin{matrix}60\\32767\end{matrix}\right\}$ /LINES = nnnn
  - 100/ERRORS = nnn
  - USLINIT
  - QUOTE = $\left\{\begin{matrix}"\\'\end{matrix}\right\}$
  - SUBPROGRAM
  - DYNAMIC
  - DEBUG
  - BOUNDS

ADDITIONAL FOR RPG ONLY

- SEG=

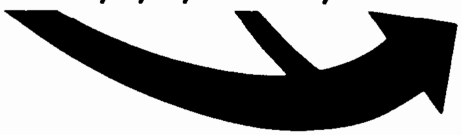
HP/3000 COMPILER COMMANDS

```
•   $ [ $ ] EDIT [VOID = seqval]  [{ SEQNUM = seqnum}]  [,INC=incnum]
                                  [{,NOSEQ          }]
```

- VOID = Bypass records (masterfile ≤ seqval) during merge
- SEQNUM = Renumber records (beginning @ seqnum) during merge
- NOSEQ = Suspend re-numbering (default) during merge
- INC = Increment records by incnum (1→999999) during merge

merging

```
:COBOL   TFILE, ,  ,MFILE,   NFILE
```



LANGUAGE AIDS

COBOL:	
HP/3000 COBOL SELF STUDY COURSE	22957A
HP/3000 COBOL REFERENCE MANUAL	32213-90001
RPG:	
IBM SYSTEM/3 TO HP/3000 CONVERSION GUIDE	32104-90004
HP/3000 RPG REFERENCE MANUAL	32104-90001
RPG LISTING ANALYZER	32104-90003
RSAM	
COMPOSER	
FORTRAN:	
DBM 1130/1800 TO HP/3000 FORTRAN CONVERSION	36995-90013
FORTRAN/3000 REFERENCE MANUAL	30000-90040
TRACE/3000 REFERENCE MANUAL	03000-90015
COMPILER LIBRARY REFERENCE MANUAL	30000-90028
SCIENTIFIC LIBRARY REFERENCE MANUAL	30000-90027
BASIC:	
HP/3000 BASIC SELF STUDY COURSE	22958A
BASIC/3000 INTERPRETER REFERENCE MANUAL	30000-90026
BASIC/3000 COMPILER REFERENCE MANUAL	32103-90001
BASIC/3000 INTERPRETER POCKET GUIDE	03000-90050
SPL:	
HP/3000 SPL REFERENCE MANUAL	30000-90024
HP/3000 SPL TEXTBOOK	30000-90025

FORTRAN EXTENSIONS/ CONSIDERATIONS

- CHARACTER DATA TYPE
- EXTENSION TO LOGICAL DATA TYPE
- ARRAYS (MAX 255 DIMENSIONS)
- EXPRESSIONS ALLOWABLE FOR SUBSCRIPTS
- DATA STATEMENT CAN USE " " OR HOLLERITH
- EXPRESSION HIERARCHY
- COMPOSITE NUMBERS
- LOGICAL OPERATORS
- PARTIAL WORD DESIGNATORS
- CHARACTER EXPRESSIONS
- DO LOOPS
- FREE FIELD FORMAT
- MEMORY TO MEMORY CONVERSION
- DISPLAY/ACCEPT
- FORMATS

basic extensions/considerations

- DATA TYPES
- FUNCTIONS
- FILES (INCLUDING FILE LOCK)
- BLOCK STRUCTURE
- RUN TIME PERFORMANCE STATISTICS
- BUILT IN EDITING
- BUILT IN DEBUGGING
- VARIABLES
- STRING HANDLING
- BUFFERED I/O
- PROMPT CAN BE INCLUDED IN INPUT STATEMENT
- SEGMENTATION BY INVOKE AND CHAIN

COBOL/3000 FEATURES

- ▶ HIGHEST LEVEL FEDERAL STANDARD COBOL
- ▶ COMMUNICATION WITH NON-COBOL LANGUAGE PROGRAMS
- ▶ DIRECT COMMUNICATION WITH SORT/3000
- ▶ DIRECT COMMUNICATION WITH IMAGE 3000
- ▶ SEGMENTED LIBRARIES THROUGH "DYNAMIC" TYPE SUBPROGRAMS

COPY LIBRARY DESCRIPTION

- Contains COBOL source statements available to a source program at compile time.

- Columns 73-80 contain the library-name to which the source statement belongs.

    - This is the library-name used in the COPY statement

    - Library-name must be left justified (if less than eight characters)

- Source statements must be syntactically correct and must not include another COPY statement.

- Source statements should be arranged in the order they are used (more efficient).

- Maintained by the user utilizing the TEXT EDITOR or FCOPY.

- The copy library file name defaults to COPYLIB or a user's library-file-name may be used by equating COPYLIB to that file-name,

    i.e. :FILE COPYLIB = user-copy-library-file-name.

NOTE: IF COPYLIB IS USED IN DATA DIVISION, FIRST RECORD IN EACH PART MUST BE A PERIOD.

```
                023400
                023500 01  MONTH-TABLE COPY MONTHTAB.
MONTHTAB                   .
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "  JANUARY".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE " FEBRUARY".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "    MARCH".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "    APRIL".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "      MAY".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "     JUNE".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "     JULY".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "   AUGUST".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "SEPTEMBER".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE "  OCTOBER".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE " NOVEMBER".
MONTHTAB                   05  FILLER          PIC X(9)  VALUE " DECEMBER".
MONTHTAB               01  MONTH-TABLE-1 REDEFINES MONTH_TABLE.
MONTHTAB                   05  MONTH           PIC X(9) OCCURS 12 TIMES.
```

COBOL/3000 CONSIDERATIONS (CONT.)

- COBOL CALLABLE INTRINSICS

  — non-supported —

  — special program —

  — typed procedures —

  — condition code —

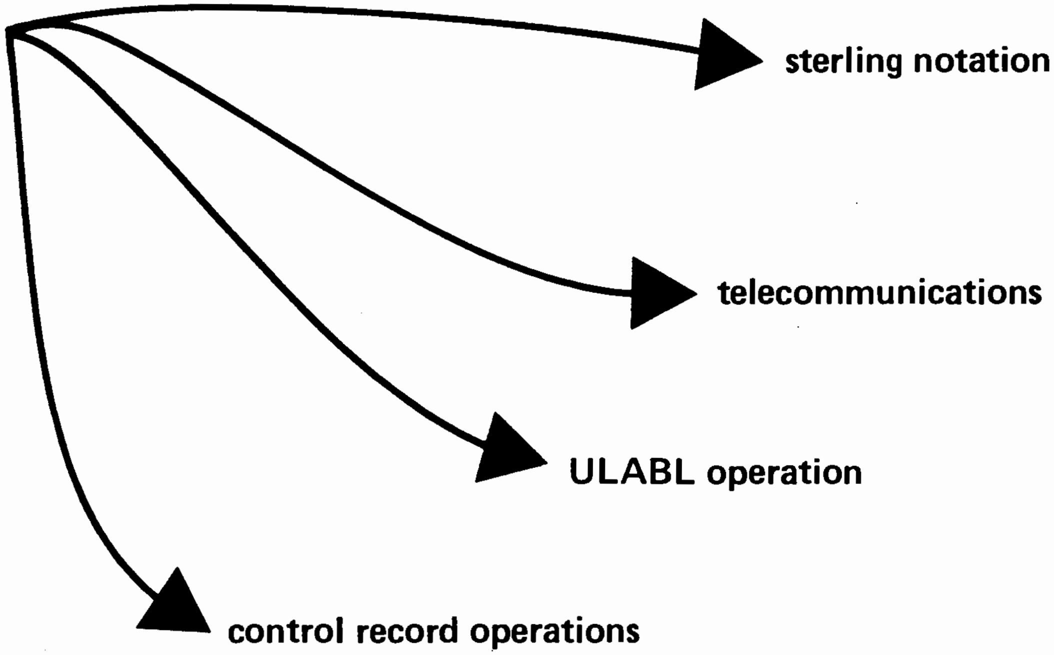
  — bit manipulation —

RPG/ 3000 FEATURES

- HP EXTENSIONS
  - — external subroutine call parameters —
  - — run-time error options —
  - — cross-reference listing option —
  - — automatic program segmentation —
  - — ebcdic/ascii translation —
  - — combined i/o (terminal) file —
  - — calculation indicator repetition —

RPG/3000 CONSIDERATIONS (CONT.)

- NON-SUPPORTED



sterling notation

telecommunications

ULABL operation

control record operations

RPG/3000 CONSIDERATIONS

- CONVERSION REQUIREMENTS

▷ INDEXED SEQUENTIAL FILES

▷ PRINTER FILES

▷ COMPILE-TIME TABLES/ARRAYS

▷ CARD READER/PUNCH/INTERPRETER

▷ EDIT WORDS

▷ ASCII vs. EBCDIC CODE

▷ DEVICE CLASS NAMES

▷ REWIND OPERATIONS

▷ QUOTATION MARKS

▷ FILE/PROGRAM NAMES

PURPOSES OF SEGMENTER SUBSYSTEM

- ALLOW USER TO MANAGE USL'S BY ADDING, DELETING ACTIVATING OR DEACTIVATING RBM'S WITHIN A USL

- MANAGE THE VARIOUS CODE LIBRARIES THAT RESOLVE EXTERNAL REFERENCES FROM THE PROGRAM

NOTE: THIS SUBSYSTEM ALLOWS RESEGMENTATING OF A PROGRAM WITHOUT RECOMPILATION

THIS PAGE ACCIDENTLY LEFT BLANK !!

CODE SEGMENTATION CONSIDERATIONS

- PROGRAM SIZE
- PROCEDURE INTERACTION
- FREQUENCY OF USE
- INSTALLATION WORK LOAD OTHER JOBS AT THE SAME TIME

software tips about the HP 3000

FCOPYing Files to Magnetic Tape

When FCOPYing a file to magnetic tape, the tape device does not rewind until the next FCOPY command is entered. If the next command does not append to the current tape file, FCOPY writes an EOF on the tape and rewinds it. Do not manually rewind or dismount the tape before entering another FCOPY command. If you do, the tape will not contain a proper EOF, and your Job/Session will wait for the tape drive to become ready so that FCOPY can write the EOF. While your Session is waiting, the terminal is locked out. If someone else mounts a tape with a write ring on your tape unit, they may find to their dismay that FCOPY has written an EOF on their good tape.

To free the terminal, mount a scratch tape with a write ring on the tape unit owned by FCOPY. If you have already entered another FCOPY command, or attempted to abort your Session, FCOPY will write an EOF on the scratch tape and rewind it. Your terminal should become available for further use. To obtain a tape with a valid EOF, re-do the previous FCOPY function(s) and allow FCOPY to rewind the tape for you.

Sam Boot
HP General Systems

SEGMENTATION FOR MAXIMUM EFFICIENCY OF SYSTEM-TYPE PROGRAMS

The purpose of this article is to describe, for the benefit of system programmers, some guidelines for the optimum design of programs for the 3000; in particular, attention will be given to the questions of segmentation.

The 3000 is a process oriented machine, incorporating the separation of code and data, and stack architecture. This permits easy design of re-entrant code. The purpose here is to discuss ways of making a particular process

a. Run as fast as possible

b. Have minimum effect on other processes in the system.

As more and more load is applied to a machine like the 3000, a point is reached where all users experience a very rapid deterioration of service. This corresponds to a kind of 'overload' condition where the system is working harder to switch from job to job than running your programs. The size of memory is the primary determinant of this point, but given a fixed memory size, the size of your programs and the quality of this segmentation have a strong influence on the work the machine will accept before overloading.

Process Environment

When you write a program, it is executed by MPE in the form shown in Figure 1. The process has a single data segment (or "stack") and a variable number of code segments of varying sizes. When you write your program you can control:

a. the size of the stack

b. the number of your code segments

c. the size of each segment

d. which code goes into which segment.

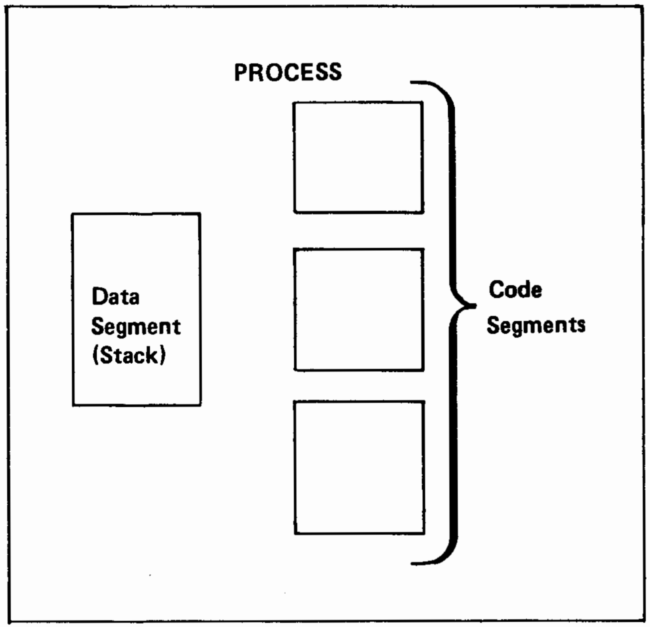


Figure 1.

The diagram shown above is actually a simplification since it does not show the externals referenced by your program (see Figure 2). If for example, your SPL-written program calls FOPEN, then a link will be created from your code to an MPE segment containing the FOPEN intrinsic code. Most of these intrinsics and all the Compiler Library routines are not in memory permanently, thus they are viewed by MPE as code segments identical to your own even though they were not written by you. For programs written in SPL, you are in control of which external procedures are called, since the calls are made explicitly. For other languages, the compiler will implicitly create in your program calls to external routines in order to perform, for example, a Fortran WRITE or a COBOL DISPLAY. The environment of a non-SPL program is harder to control because it requires a knowledge of when the compiler will

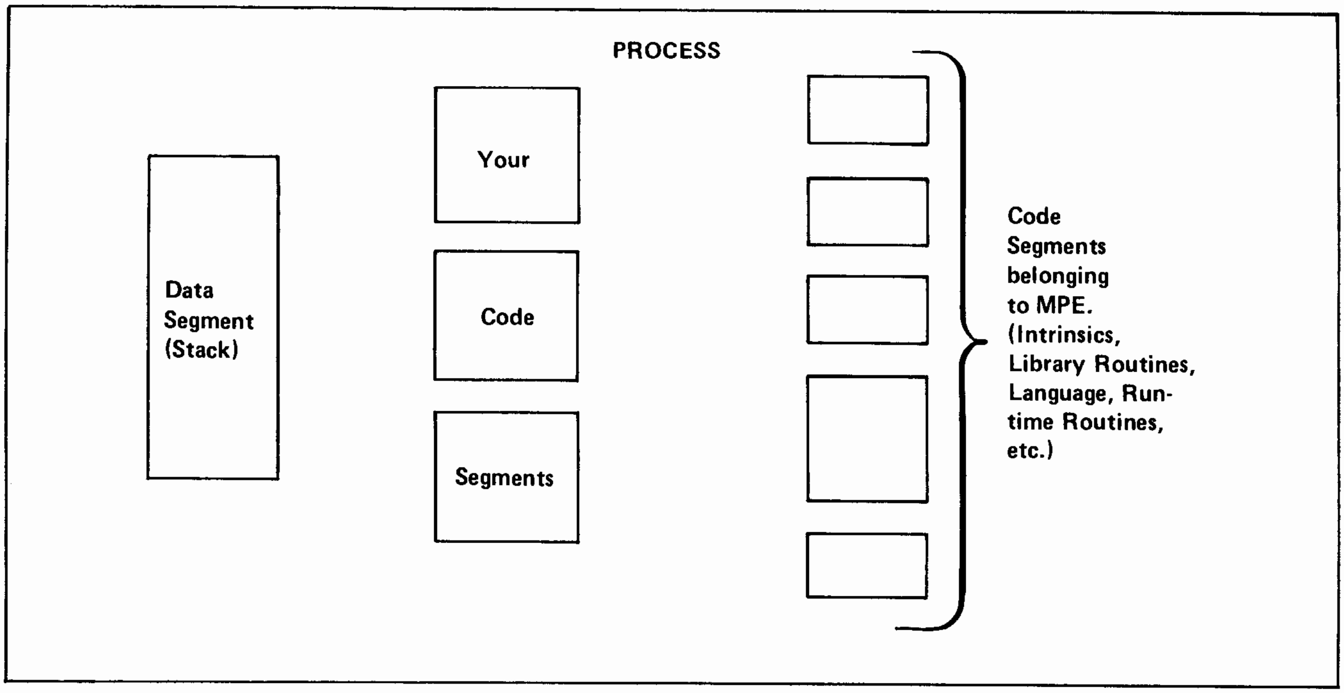

PROCESS
Data Segment (Stack)
Your
Code
Segments
Code Segments belonging to MPE. (Intrinsics, Library Routines, Language, Run-time Routines, etc.)

Figure 2.

emit those external calls. We will limit this discussion to those areas over which you have primary control: your own program code and data stack. Given any language, there are some fundamental principles to follow which will decrease the run-time of a process and its impact on system load.

How to Determine a Program Environment

When you prepare your program the PMAP option will show you the size of each segment, which procedures are in which segment, and the names of externals called by each segment. The MPE manual describes the format of the PMAP in detail.

How MPE Runs Your Program

There are two MPE modules concerned here – the dispatcher and the memory management system. The dispatcher is responsible for the allocation of CPU time to all the executing processes. The memory management system has the job of fitting code and data segments into memory as they are required, this operation often requiring the decision of which segment(s) to delete to make space. When your time-slice starts, the stack is made present in memory and control is passed to the program. As the program proceeds, it will call procedures which are not in the current segment. At this point your program is suspended while MPE arranges to make the required segment present. This can take from 20 to 100 milliseconds since a disc access is involved. While this is going on the dispatcher tries to run the process with the next highest priority which is already resident in memory. When the destination segment has been made present, control is passed to the procedure originally called.

The point to note here is that calling a procedure in an absent code segment is a time-consuming job.

How Do I Tell If A Segment Will Be Present?

You can't for sure. The memory management system will attempt to keep the most popular segments in memory, and the system is aware, using an internal table, which segments you use most in your process. Using this information the system will postpone for as long as possible disrupting your process, but in a busy system it is very difficult to predict the state of memory.

Rules for Segmenting Your Program

Rule No. 1

Minimize the number of times the program crosses a segment boundary. In other words, stay within a segment for as long as possible. When you leave it, stay out for as long as possible.

Design of Programs is Important

Do not leave segmentation to the last minute. As will be shown below, it is possible to write programs that cannot be correctly segmented.

Any procedure or outer block Relocatable Binary Module (RBM) must reside wholly within a segment. Thus if it proves necessary to move a block of code into a separate segment, it will only be possible if the code is a procedure. You cannot take an arbitrary set of instructions and place them into a named segment – the whole RBM must be moved. Therefore, the way you divide your program into procedures is vitally important in the design phase.

Concept of Locality

The locality of a program is the degree to which control remains in the same general area of code. A high locality means that control remains in the same area for a long period of time. Poor locality means the program branches wildly and rapidly all over the place. The 3000 needs programs that have good segment locality but does not care about the degree of locality within any given segment. That is to say, it does not want programs that jump from segment to segment continuously but once inside any given segment, it doesn't matter what the locality is like.

Functional vs. Temporal Segmentation

Intuitively, one segments according to the function of the procedures. That is, all the command decoding routines are put together, the command executors are put together, etc. This is wrong. Wrong. Segmentation is a speed-enhancing operation thus time, not function, is the critical dimension. Since Rule No. 1 says stay inside a segment for as long as you can, control must pass smoothly from segment to segment as the program progresses.

As an example, consider a small utility program which dumps a file to the line printer in some special format. Let us suppose that the operator can choose the name of the file and which of three possible formats to use. The program is written with four procedures: A, B, C, and D.

Let us further suppose that each dump routine has a procedure to fetch a record from its file and a procedure to format a print line:

It would be tempting to put all the formatting routines in one segment, and the record fetching routines in another. This would cause a segment boundary to be crossed twice for every record dumped – perhaps a thousand times. The correct way is to put B1B2 together, C1C2, etc. If A is in its own segment then only three segment boundaries are crossed for a whole dump. In a busy system this simple change could make large differences in the run time of your program.

To sum up, estimate the number of times a segment boundary is crossed in your program and multiply this by 40 milliseconds (12 msec if you have a swapping disc and your program resides on it). This is the time your program will be doing no useful work and other processes will be disrupted.

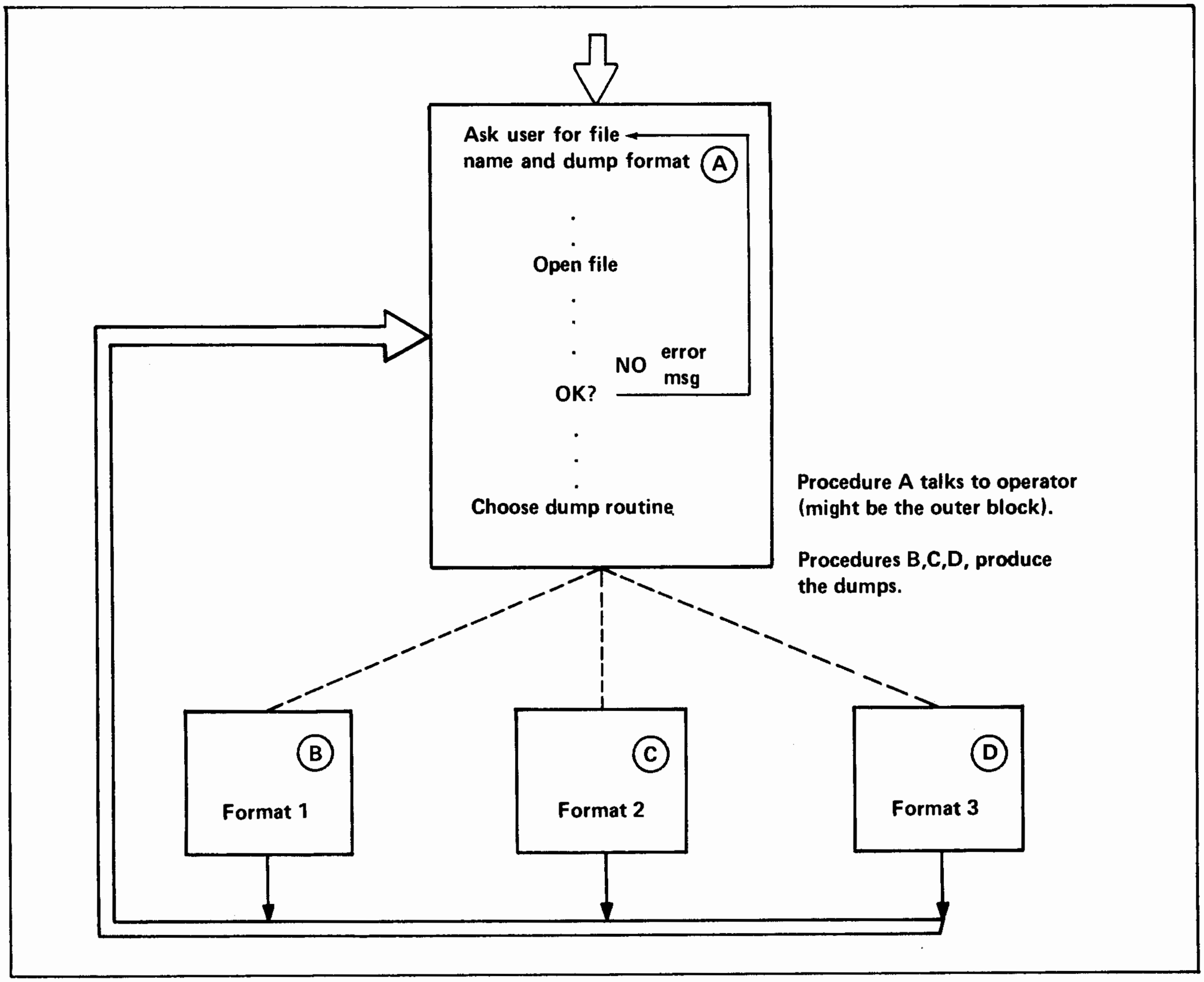


Figure 3.

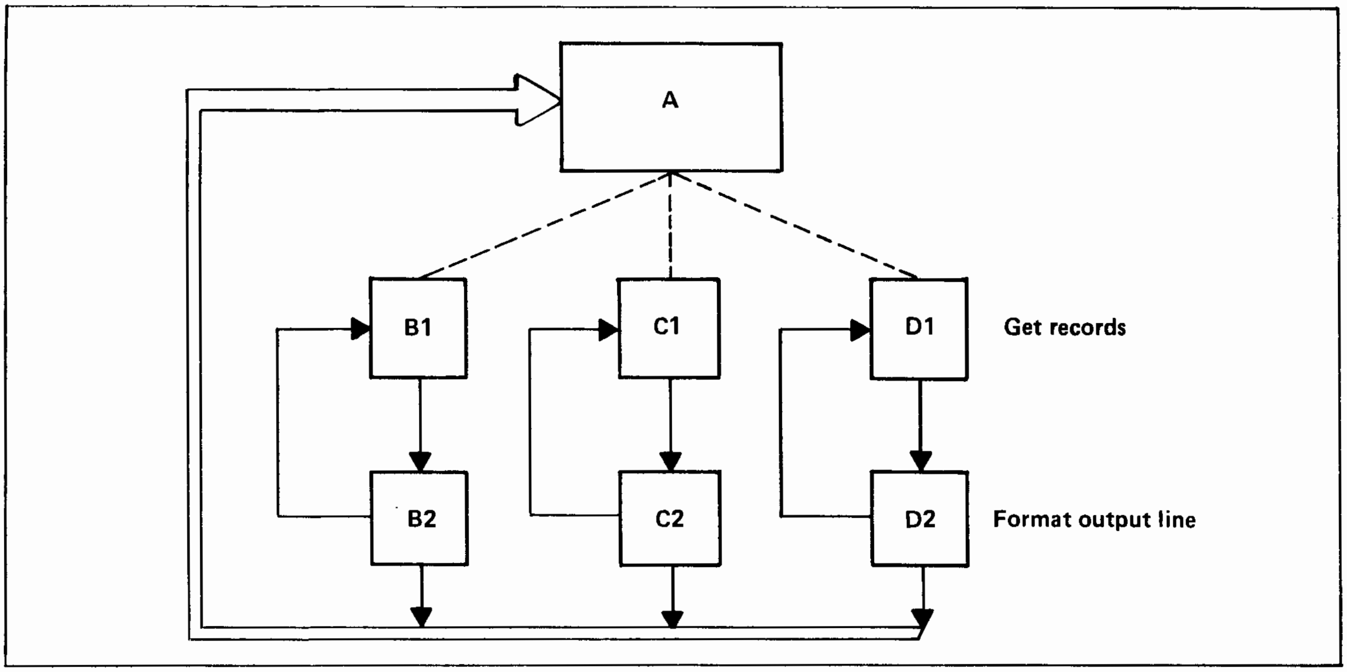


Figure 4.

Rule No. 2

Do Not Burden Your Working Set With Infrequently Used Code

Let us suppose that you have arrived at some segmentation scheme using the above rule so that you have good segment locality. The next step is to reduce the size of the 'working set'.

Frequency of Code Use

The 'working set' of segments is the set that consumes most of the CPU time. For example in the program above the working set is the code that executes the main loop such as C1C2. Let us assume that C1C2 are in a segment of their own called CSEG. The system may spend minutes in this segment for a large dump. It is important therefore to minimize its size in order to reduce contention for the scarce memory resource.

To do this, examine the codes in the working set and remove any code that exectues infrequently. Very often, this applies to code which does error-handling. When your program detects an error, do not handle it in-line. Write an error-message generating procedure and call it with a parameter indicating which message to output. This can be put in a separate segment and thus not clutter up memory while doing normal error-free processing. As another example, suppose that in the program mentioned above, after doing an FWRITE, you check the condition code for end-of-file and, if required, execute a somewhat elaborate sequence to extend the file by building a new one and copying the old into it and then purging the old file. If this condition is likely to occur once in every 500 runs, why hold it in precious memory with the working set? Banish it to some auxiliary segment and let MPE bring it in only when needed. Remember that you can only move this code if it is a procedure.

```
WRONG                    RIGHT

FWRITE(. . .);           FWRITE(. . .);
IF>THEN                  IF>THEN EXTEND'
BEGIN                    FILE;
.
.
<<LENGTHEN FILE>>        Procedure EXTEND'FILE
.                        is put in another segment
.
END;
```

Segment Sizes

This is a trade-off. If you segment into many small segments, each one has to be separately read into memory before your program can begin execution at the start of a time-slice. (Segments are in fact only read in when actually referenced, but a program with dozens of small segments is likely to need several of them before any real work can be done). This leaves less of the time-slice for useful work.

At the other end of the spectrum, a program with a few large segments will take up a lot of memory — perhaps unnecessarily. Segments should be typically around 3K decimal words, but if you have lots of memory and are nowhere near the machine overload point, larger segments may enhance throughput slightly. Such a strategy may cause trouble later however when machine load increases.

How Many Segments

As a guide, a program is getting large at 10 segments or so. A typical compiler has around 25 while a small utility like SORT has about 3. There is a hardware limitation of 63 code segments in a process.

Rule No. 3

Make segment as small as possible with a maximum of about 3K decimal words.

Rule No. 4

If Rule 3 has to be violated in order to reduce the number of segments, keep principal working sets small and make infrequently used segments large.

If Your Code is Shared

If your program is going to be run from multiple terminals then the code segments will automatically be shared by the multiple processes. Each process will have its own stack of course. If your program design requires data which is never altered such as error messages, look-up tables, etc., then by placing them in the code rather than the stack, only one copy is required for all processes.

WRONG

```
BEGIN
BYTE ARRAY MESSG (0:22):="TOO
MANY TIMES ENTERED";                Global Declara-
                                    tions

.
.
.
.
PROCEDURE MESSOUT;    Procedure to print error
BEGIN                 message
PRINT (MESSG,-23,0);
:
END;
  :
END.
```

WHY WRONG? The array MESSG is present in the stack perpetually. Each process running this program carries the message string around in its stack.

RIGHT

```
BEGIN          MESG only exists while MESSOUT
.              executes.  SPL will store the string
.              in quotes in the code segment –
.              effectively making it shared.  The
.              stack is now smaller.
.
.
PROCEDURE MESSOUT;
BEGIN
BYTE ARRAY MESG(0:22);
MOVE MESG:="TOO MANY VALUES ENTERED";
PRINT (MESG,-23,0);
END;
.
.
.
END.
```

Rule No. 5

In SPL, keep initialized variables, especially arrays, out of the GLOBAL DECLARATIONS.

In Fortran, infrequently used variables and arrays should not be initialized in DATA statements.

SEGMENTER

RBM

PROGRAM UNIT COMPILED FROM SOURCE CODE
AND RESIDING IN A USL FILE

BASICCOMP

- PROGRAM

FORTRAN

- MAIN PROGRAM
- SUBROUTINE
- FUNCTION ROUTINE

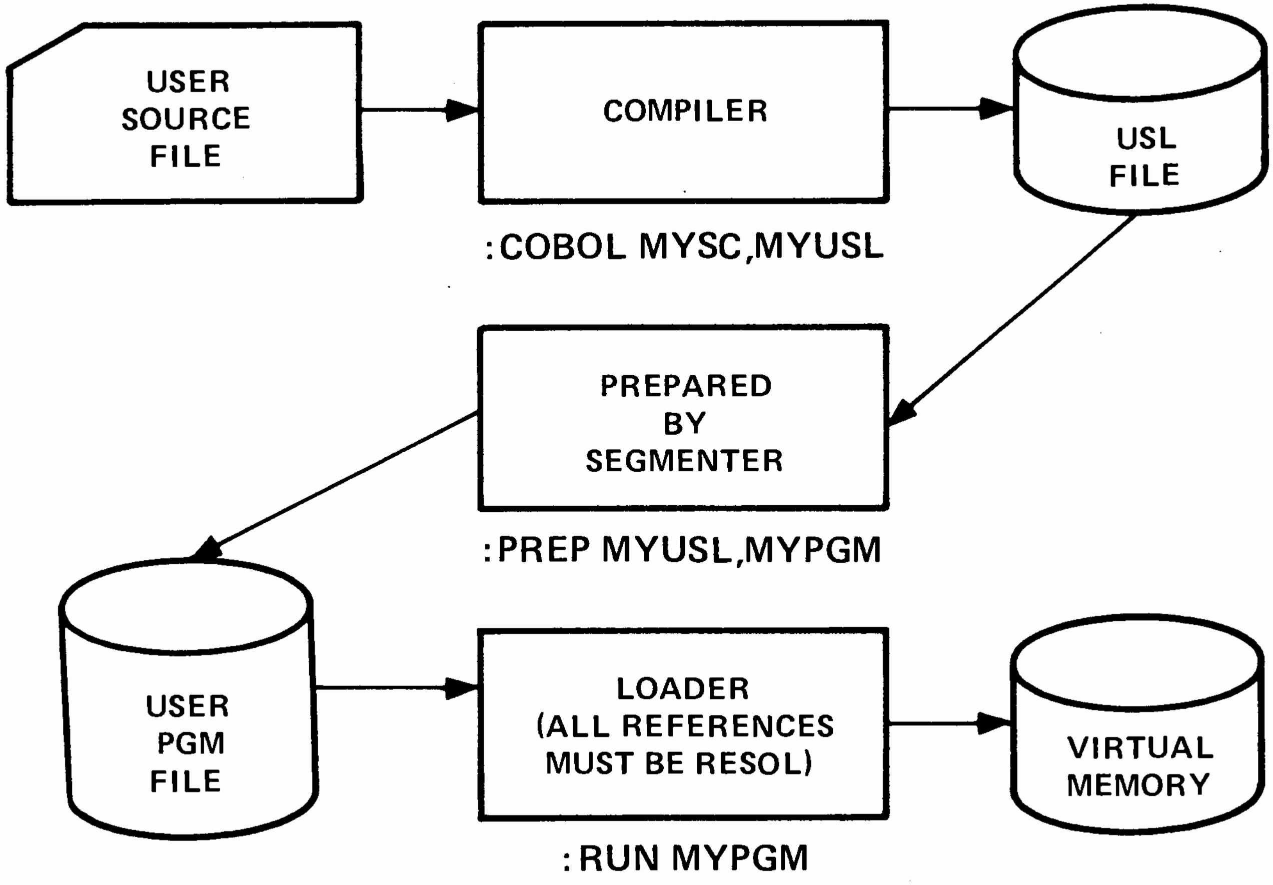
RPG

- MAIN PROGRAM
- SUBROUTINES

COBOL

- MAIN PROGRAM
- SECTIONS
- SUBPROGRAMS

SEGMENTER (CONT.)



USER SOURCE FILE

COMPILER

USL FILE

:COBOL MYSC,MYUSL

PREPARED BY SEGMENTER

:PREP MYUSL,MYPGM

USER PGM FILE

LOADER (ALL REFERENCES MUST BE RESOL)

VIRTUAL MEMORY

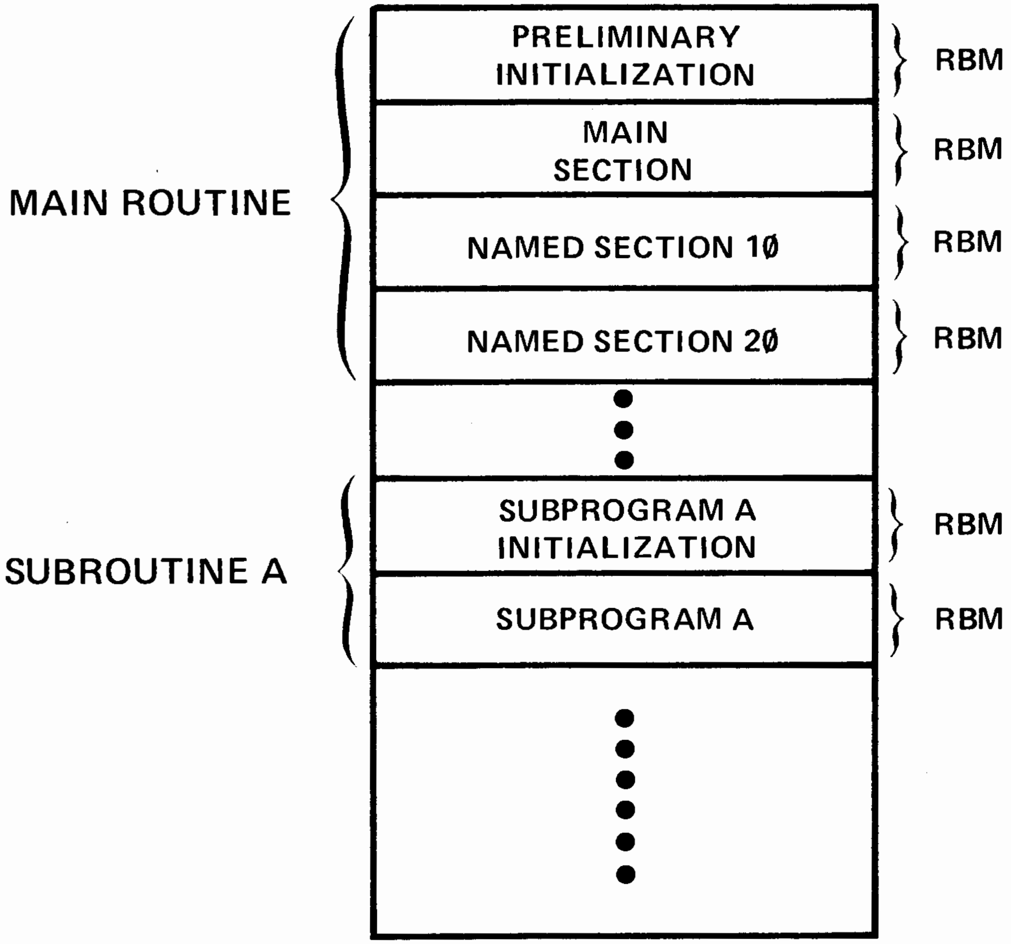
:RUN MYPGM

CONTROL OF SEGMENTATION AT COMPILE TIME

```
:FORTRAN,MYUSL
[$CONTROL SEGMENT = SEG1]
PROGRAM NAM1
I = 1
STOP
END
[$CONTROL SEGMENT = SEG2]
   FUNCTION ABC(X)
   INTEGER X
   ABC = X
   RETURN
   END
```

SEGMENTER (CONT.)

COBOL PROGRAM



SEGMENTER (CONT.)

COBOL EXAMPLE

①

```
   HEWLETT-PACKARD 32213A.01.2

        $CONTROL USLINIT,SOURCE
        $EDIT SEQNUM=100
 000100*********************** 
 000200*     THIS PROGRAM CONTAIN
 000300*          DESCRIBED AT THE
 000400*          THE HP-3000 COBO
 000500***********************
 000600
 000700*********** I D E N T I F
 000800
 000900 IDENTIFICATION DIVISION.
—001000 PROGRAM-ID.
 001100      EMPLOYEE-REPORT.
 001200 AUTHOR.
 001300      ORLAND LARSON, COMPU
 001400 INSTALLATION.
 001500      DATA SYSTEMS DIVISION
 001600 DATE-WRITTEN.
 001700      JUNE 1974.
 001800 DATE-COMPILED.
```

②

```
 023900*************** P R O C E D U R E   D I V I S I O N ***************
 024000
 024100 PROCEDURE DIVISION.
 024200
 024300***************************************************************
 024400************************ SORT EXAMPLE ************************
 024500***************************************************************
 024600
—024700 SORT-SECTION SECTION 10.
 024800 SORT-PROCEDURE.
 024900      DISPLAY "EXECUTION BEGAN AT " TIME-OF-DAY.
 025000      SORT SORT-FILE
 025100           ON ASCENDING KEY SEX-IN
 025200           ON DESCENDING KEY YRS-OF-SERV-IN
 025300           ON ASCENDING KEY AGE-IN
 025400           USING EMPLOYEE-FILE
 025500           OUTPUT PROCEDURE MAIN-SECTION.
 025600      STOP RUN.
 025700
—025800 MAIN-SECTION SECTION 20.
 025900 BEGIN.
 026000      OPEN OUTPUT PRINT-FILE.
```

③

```
        $CONTROL SUBPROGRAM,SOURCE
        $EDIT SEQNUM=100
 000100
 000200***************************************************************
 000300******** THIS IS A COBOL SUBPROGRAM TO COMPUTE AVERAGES. ********
 000400***************************************************************
 000500
 000600 IDENTIFICATION DIVISION.
—000700 PROGRAM-ID. AVERGEC.
 000800 ENVIRONMENT DIVISION.
 000900 DATA DIVISION.
 001000 LINKAGE SECTION.
 001100 77  AVERAGE  PIC S99V9 USAGE IS COMP.
 001200 77  DIVIDEND PIC S9(4) USAGE IS COMP.
 001300 77  DIVISOR  PIC S9(3) USAGE IS COMP.
 001400 PROCEDURE DIVISION USING AVERAGE, DIVIDEND, DIVISOR.
 001500 CALCULATIONS.
 001600      COMPUTE AVERAGE = DIVIDEND / DIVISOR.
 001700      GOBACK.
```

SEGMENTER (CONT.)

COBOL PMAP EXAMPLE

```
:PREP EMPUSL,EMPROG;STACK=10000:PMAP
PROGRAM FILE EMPPROG.COBOL.TRAINING
EMPLOYEEREPO          0
    NAME                   STT    CODE     ENTRY    SEG
    EMPLOYEEREPO'            1       0         0             PRELIMINARY
    DEBUG                    2                        ?   -  INITIALIZATION
    SORTSECTION10'           3                        2      SEGMENT
    MAINSECTION20'           4                        1
    C'GOTO                   5                        ?
    SEGMENT LENGTH ________________ [60]
MAINSECTION20'        1
    NAME                   STT    CODE     ENTRY    SEG
    MAINSECTION20'           1       0         0
    C'OPEN                   2                        ?
    C'ENDPAR                 3                        ?
      •                      •                        •   -  MAIN
      •  21 entries omitted  •                        •      SECTION
      •                      •                        •
    TERMINATE'              31                        ?                      EMPLOYEE
    SEGMENT LENGTH ______________ [1410]                                 -   REPORT
SORTSECTION10'        2
    NAME                   STT    CODE     ENTRY    SEG
    SORTSECTION10'           1       0         0
    C'DISPLAY'INIT           2                        ?
    C'DISPLAY'L              3                        ?
    C'DISPLAY'SR             4                        ?
    C'DISPLAY'FIN            5                        ?
    C'ASSIGNPHASE            6                        ?      SORT
    ASSIGNPHASE              7                        ?   -  SECTION
    C'CLOSE                 10                        ?
    C'OPEN                  11                        ?
    C'INPUTPHASE            12                        ?
    SORTPHASE               13                        ?
    C'PERFORM               14                        ?
    FINALEXITPHASE          15                        ?
    TERMINATE'              16                        ?
    SEGMENT LENGTH ________________ [124]
AVERGEC'              3
    NAME                   STT    CODE     ENTRY    SEG
    AVERGEC'                 1       0         0
    AVERGEC                  2                        4
    SEGMENT LENGTH ________________ [70]
AVERGEC               4
    NAME                   STT    CODE     ENTRY    SEG  -   COBOL
    AVERGEC                  1       0         0             SUBPROGRAM
    AVERGEC'                 2                        3
    C'LOAD                   3                        ?
    C'DIVIDE                 4                        ?
    C'STORE                  5                        ?
    SEGMENT LENGTH ________________ [74]

PRIMARY DB          0       INITIAL STACK    23420      CAPABILITY       600
SECONDARY DB     3637       INITIAL DL           0      TOTAL CODE      2000
TOTAL DB         3637       MAXIMUM DATA         ?      TOTAL RECORDS     40
ELAPSED TIME    00:00:11.136                     PROCESSOR TIME    00:02.478
```

SPL PMAP EXAMPLE

```
PROGRAM FILE XP.PUB.Z                                  $CONTROL SEGMENT = MAIN

MAIN            0
   NAME             STT   CODE  ENTRY  SEG             SPL GENERATED
   OB                1     0      0
   DBOPEN            2                  ?
   DBFIND            3                  ?
   DBGET             4                  ?
   FOPEN             5                  ?
   PRINT'FILE'INFO   6                  ?
   GOTY              7                  4
   XCONTRAP         10                  ?
   PRINT            11                  ?
   INPUT            12                  4
   TERMINATE        13                  ?
   SCANNER          14                  4
   SELECT           15                  2
   SHOW             16                  2
   DBLK             17                  4
   LEN              20                  4
   EXP              21                  1
   BINARY           22                  ?
   ERROR            23                  3
   TERMINATE'       24                  ?
   SEGMENT LENGTH          764
EXPLODE         1                                      $CONTROL SEGMENT = EXPLODE
   NAME             STT   CODE  ENTRY  SEG
   EXP               1     0      0                    PROCEDURE EXP (. . .);
   SHOWY             3                  4
   XCONTRAP          4                  ?
   RESETCONTROL      5                  ?
   CLR               6                  4
   SCANNER           7                  4
   BINARY           10                  ?
   ERROR            11                  3
   ASCII            12                  ?
   FOPEN            13                  ?
   PRINT'FILE'INFO  14                  ?
   FCLOSE           15                  ?
   PRINT            16                  ?
   DBFIND           17                  ?
   DBGET            20                  ?
   GOTY             21                  4                  PROCEDURE MOVER (. . .);
   MOVER             2    2631   2631
   DMPY'            22                  ?
   FWRITE           23                  ?
   TERMINATE        24                  ?
   CLRW             25                  4
   DASCII           26                  ?
   LEN              27                  4
   SEGMENT LENGTH         3654
SELECT          2                                      $CONTROL SEGMENT = SELECT
   NAME             STI   CODE  ENTRY  SEG
   SHOW              1     0      0                    PROCEDURE SHOW (. . .);
   GOY               3                  4
   SCANNER           4                  4
   BINARY            5                  ?
   FOPEN             6                  ?
   PRINT'FILE'INFO   7                  ?
```

SPL PMAP EXAMPLE (CONT'D)

```
    GOTY              10                4
    XCONTRAP          11                ?
    ERROR             12                3
    ASCII             13                ?
    FWRITE            14                ?
    CLRW              15                4
    LEN               16                4
    FCLOSE            17                ?
    SELECT             2   1531   1531                PROCEDURE SELECT (. . .);
    TRUNC             20                3
    CLR               21                4
    SEGMENT LENGTH         3070
ERROR               3                                $CONTROL SEGMENT = ERROR
    NAME             STT   CODE  ENTRY  SEG
    TRUNC              1      0      0
    PRINT              3                ?
    ERROR              2     36     36                PROCEDURE ERROR (. . .);
    CLRW               4                4
    ASCII              5                ?
    LEN                6                4
    TERMINATE          7                ?
    SEGMENT LENGTH          754
UTILITY             4                                $CONTROL SEGMENT = UTILITY
    NAME             STT   CODE  ENTRY  SEG
    INPUT              1      0      0                PROCEDURE INPUT (. . .);
    ERROR             12                3
    RESETCONTROL      13                ?
    PRINT             14                ?
    READ              15                ?
    SCANNER            2    210    210
    DBLK               3    371    371                      PROCEDURE . . . .
    SHOWY              4    420    420
    ASCII             16                ?
    GOY                5    466    466
    GOTY               6    506    506
    LEN                7    516    516
    CLRW              10    533    533
    CLR               11    545    545
    SEGMENT LENGTH          600                                        (IA,BA)

PRIMARY DB       115    INITIAL STACK    23420    CAPABILITY         600
SECONDARY DB    1663    INITIAL DL           0    TOTAL CODE       11504
TOTAL DB        2000    MAXIMUM DATA         ?    TOTAL RECORDS       64
ELAPSED TIME   00:00:12.820                       PROCESSOR TIME  00:03.445
```

STACK = 10000

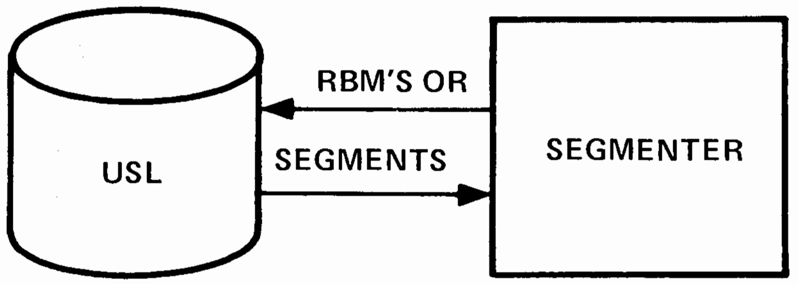
TOTAL RECORDS IN PROGRAM FILE

TOTAL LENGTH OF ALL SEGMENTS

VARIABLES DECLARED IN OUTER BLOCK CAUSES ALLOCATION OF PRIMARY DB

SPACE ALLOCATED BY DECLARATION OF ARRAYS IN OUTER BLOCK

SEGMENTER (CONT.)



USL

RBM'S OR

SEGMENTS

SEGMENTER

INVOKING SEGMENTER

```
[: FILE LP;DEV = LP]
 :SEGMENTER [*LP]
```

SEGMENTER (CONT.)

SEGMENTER USL MANAGING COMMANDS

- – BUILDUSL
- – USL
- – CEASE } change activity status
- – USE
- – NEWSEG change seg name
- – PURGERBM bury inactive RBM's
- – LISTUSL seg names etc. in a given USL
- – AUXUSL specify auxiliary library
- – COPY from one USL to another

SEGMENTER (CONT.)

SEGMENTER PREPARE COMMAND

- THE SEGMENTER CAN PREPARE A USL INTO A PROGRAM FILE

  – PREPARE   PFNAME ;OPTIONS (REF MPE 7-12)

- NO USL IS MENTIONED IN THE PREPARE COMMAND. APPROPRIATE USL MUST BE INVOKED WITH USL COMMAND.

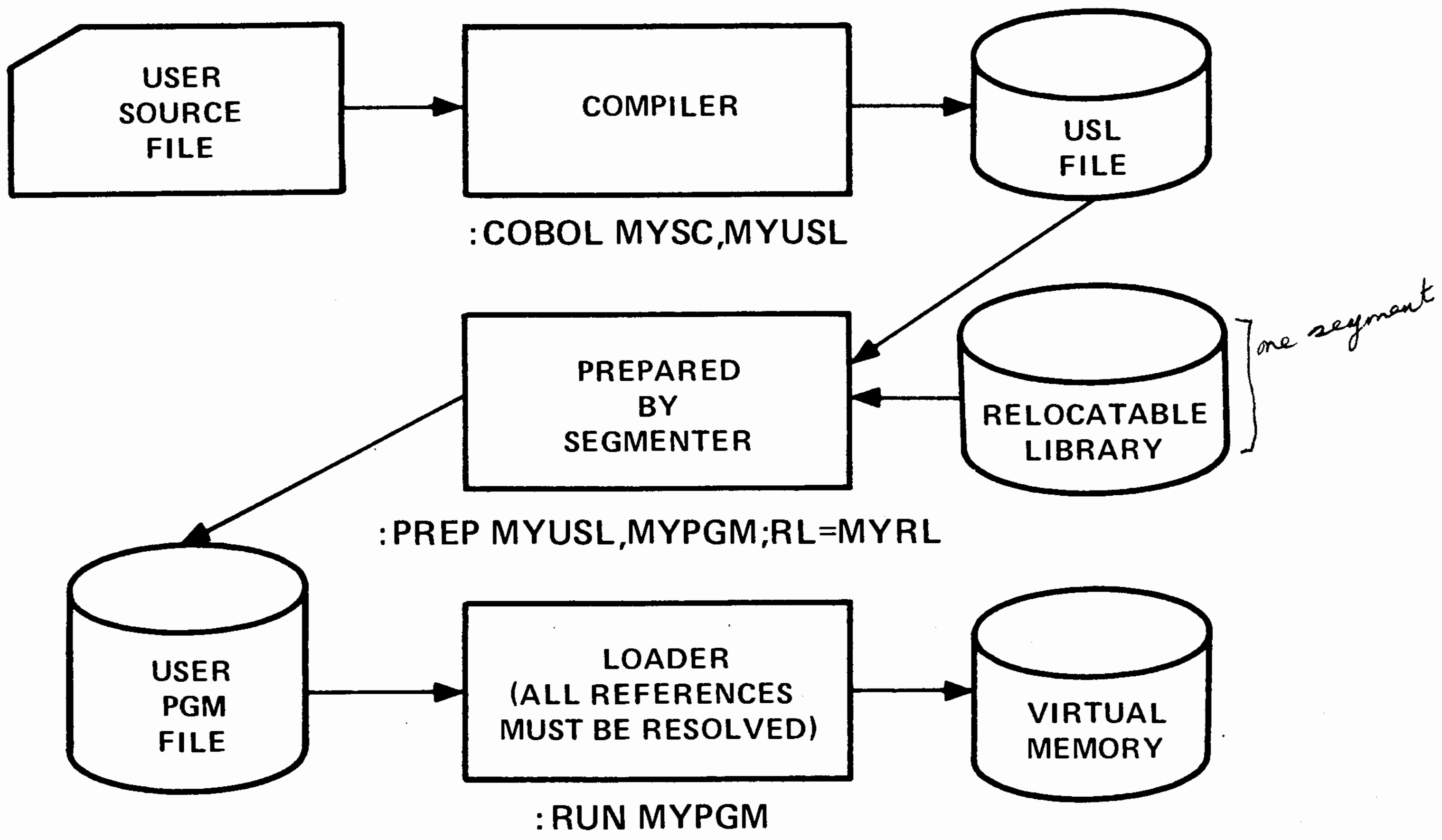
- IF PFNAME IS NOT AN OLD FILE A NEW TEMPORARY FILE IS SET UP.

- EG.

```
:SEGMENTER
—USL MYUSL
—PREPARE MYPGM
—EXIT
```

THE ABOVE SEQUENCE OF COMMANDS PREPS THE USL FILE, MYUSL, INTO THE PROGRAM FILE MYPGM.

SEGMENTER (CONT.)



```
:COBOL MYSC,MYUSL
```

```
:PREP MYUSL,MYPGM;RL=MYRL
```

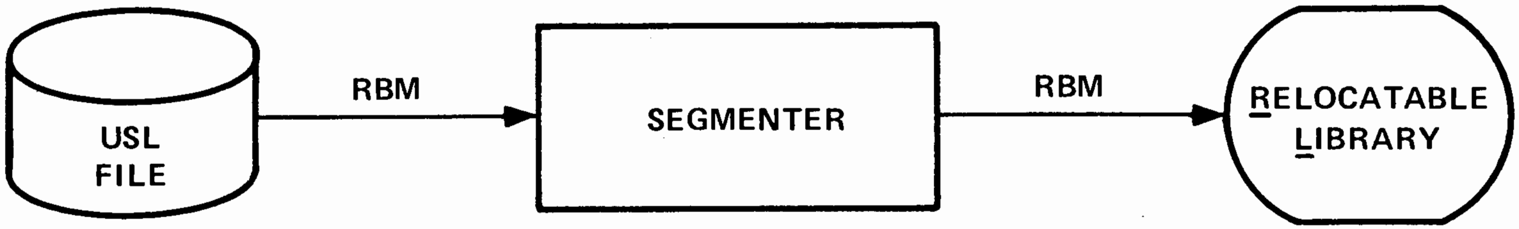
```
:RUN MYPGM
```

SEGMENTER (CONT.)

RELOCATABLE LIBRARIES

- COLLECTIONS OF RBMS
  - SEARCHED AT PREP TIME
    - PRODUCES A SINGLE UNIQUE (NON-SHARABLE) SEGMENT
      - RL ALWAYS A PERMANENT FILE
        - DOES NOT HAVE TO BELONG TO USER'S LOG-ON GROUP

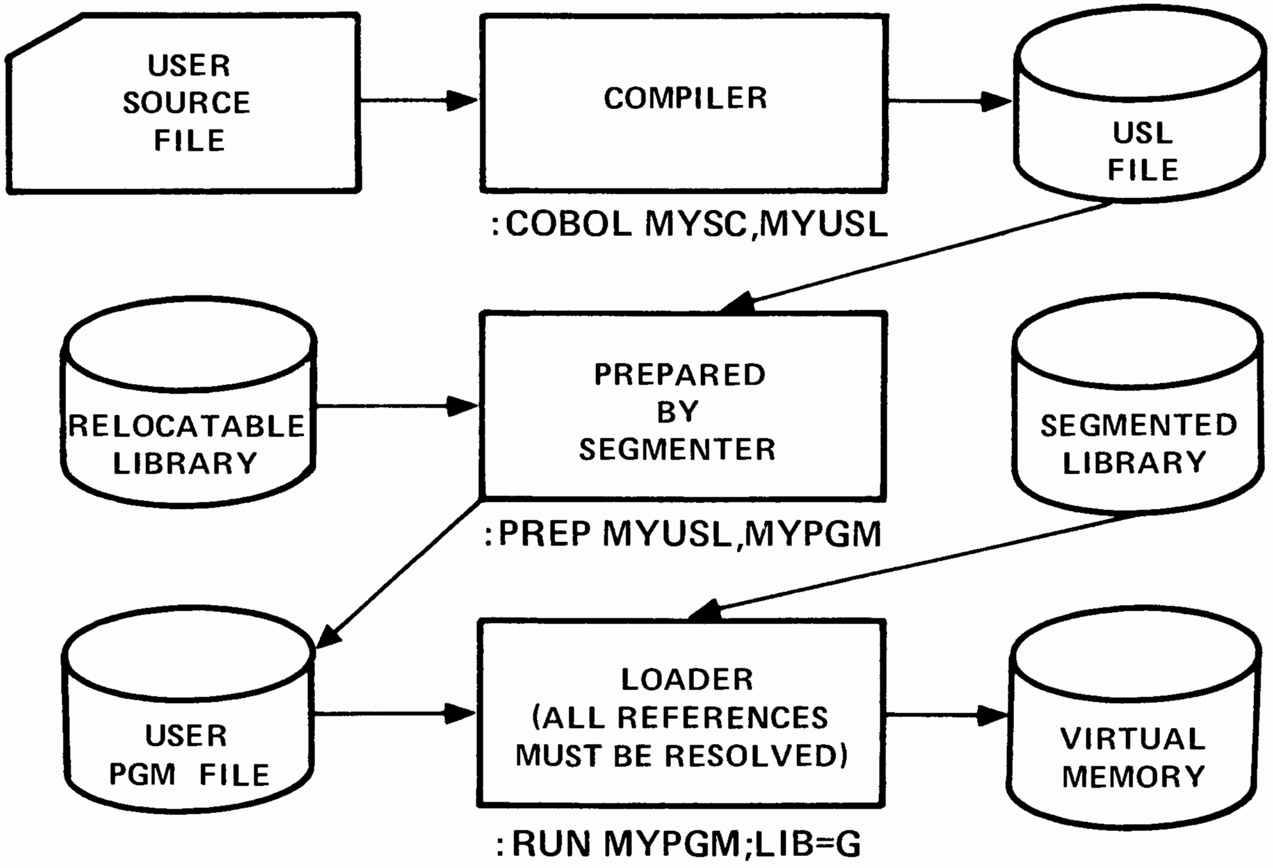
SEGMENTER (CONT.)



<u>RL'S</u>

```
—BUILDRL filename,records,extents

—RL  filename

—ADDRL  name  [,(index) ]

                 ┌ENTRY,┐
—PURGERL         │      │          name
                 └UNIT, ┘

—LISTRL
```

SEGMENTER (CONT.)



USER SOURCE FILE

COMPILER

USL FILE

:COBOL MYSC,MYUSL

RELOCATABLE LIBRARY

PREPARED BY SEGMENTER

SEGMENTED LIBRARY

:PREP MYUSL,MYPGM

USER PGM FILE

LOADER (ALL REFERENCES MUST BE RESOLVED)

VIRTUAL MEMORY

:RUN MYPGM;LIB=G

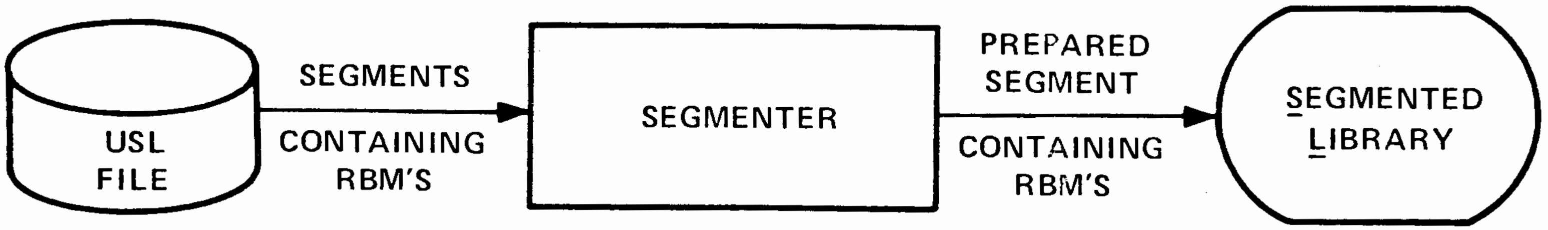
CSU 7-28

SEGMENTER (CONT.)

SEGMENTED LIBRARIES

- COLLECTIONS OF SEGMENTS IN "PREPED" FORM
  - SEARCHED AT LOAD TIME
    - ONE OR MULTIPLE (SHARABLE) SEGMENTS ARE LOADED
      - SL ALWAYS A PERMANENT FILE
        - ONE TO THREE SL'S MAY BE SEARCHED AT ONCE
          - SL'S TO BE SEARCHED, MUST HAVE THE NAME "SL" AS FILENAME
            - ONLY LOCAL VARIABLES ALLOWED

SEGMENTER (CONT.)



SL'S

—BUILDSL SL[.group],numrec,numextents

LMAP EXAMPLE

:FILE LOADLIST;DEV=LP   **:RUN DELCOBL;LIB=G;LMAP**

```
PROGRAM FILE DELCOBL.HP32206.SUPPORT

CCLOSETERM        PROG 2  31   1 GSL  0  47   0
CNEXTEDIT         PROG 2  30   1 GSL  0  53   0
CGETFORM          PROG 2  27   1 GSL  0  55   0
CFINDFORM         PROG 2  24   1 GSL  0  57   0
CEDITFIELD        PROG 2  23   1 GSL  0   2   0
CSHOWFORM         PROG 2  22   1 GSL  0   4   0
CREADTERM         PROG 2  20   1 GSL  0  41   0
CWRITETERM        PROG 2  15   1 GSL  0  44   0
CTERMSTATUS       PROG 2  14   1 GSL  0  37   0
CCLOSEFORM        PROG 2   6   1 GSL  0  61   0
COPENTERM         PROG 2   5   1 GSL  0  51   0
COPENFORM         PROG 2   3   1 GSL  0  63   0
QUIT              PROG 0  32   1 SSL  0  14 217
C'TST'            PROG 0  26   1 SSL  0   1  62
C'WRITE'          PROG 0  25   1 SSL  0   3   4
C'ENDPAR          PROG 0  21   1 SSL  0  14  27
C'PERFORM         PROG 0  16   1 SSL  0   6  27
TERMINATE'        PROG 0  13   1 SSL  0   2 140
C'DISPLAY'FIN     PROG 0  12   1 SSL  0  20   4
C'DISPLAY'ID      PROG 0  11   1 SSL  0  17   4
C'DISPLAY'L       PROG 0  10   1 SSL  0  15   4
C'DISPLAY'TLIT    PROG 0   7   1 SSL  0  16   4
C'LIT'FIG'MOVE    PROG 0   4   1 SSL  0  10  27
C'OPEN'           PROG 0   2   1 SSL  0  10   4
C'GOTO            PROG 0  17   1 SSL  0  13  27
                           5   0
COBOLTRAP         PROG 0   3   0 SSL  0  10   0
DEBUG             PROG 0   2   0 SSL  0   1 213
FREADDIR          GSL  0  77   0 SSL  0   1 221
FOPEN             GSL  0  76   0 SSL  0   1  10
FGETINFO          GSL  0  75   0 SSL  0   3  45
FCLOSE            GSL  0  74   0 SSL  0   1   7
FCONTROL          GSL  0  73   0 SSL  0   3  56
FSETMODE          GSL  0  72   0 SSL  0   2  56
ASCII             GSL  0  71   0 SSL  0  15 126
FREAD             GSL  0  70   0 SSL  0   4 221
FCHECK            GSL  0  67   0 SSL  0   4  45
FWRITE            GSL  0  66   0 SSL  0   3 221
BINARY            GSL  0  65   0 SSL  0  17 126

301 302
```

PARAMETER CHECKING OF CALLING PROCEDURE

SEGMENT NUMBER OF EXTERNAL

STT NUMBER OF SL PROCEDURE

SEGMENT NUMBER OF SL PROCEDURE

PARAMETER CHECKING OF CALLED PROCEDURE

CST NUMBERS FOR LOADED SEGMENTS

PROG OR SL

STT OF EXTERNAL

S = SYSTEM
P = PUBLIC
G = GROUP

SEGMENTER (CONT.)

SL COMMAND

- SPECIFIES A SEGMENTED LIBRARY TO BE USED AS THE OBJECT OF THE ADDSL, PURGESL, AND LISTSL COMMANDS

- THE SL SPECIFIED MAY ONLY BE ONE OF 3 FORMS:

1. THE SYSTEM LIBRARY SL

   –SL SL. PUB. SYS

2. THE ACCOUNT LIBRARY SL

   –SL SL. PUB. MYACCT

3. THE GROUP LIBRARY SL

   –SL SL. MYGROUP. MYACCT

SEGMENTER (CONT.)

ADDSL COMMAND

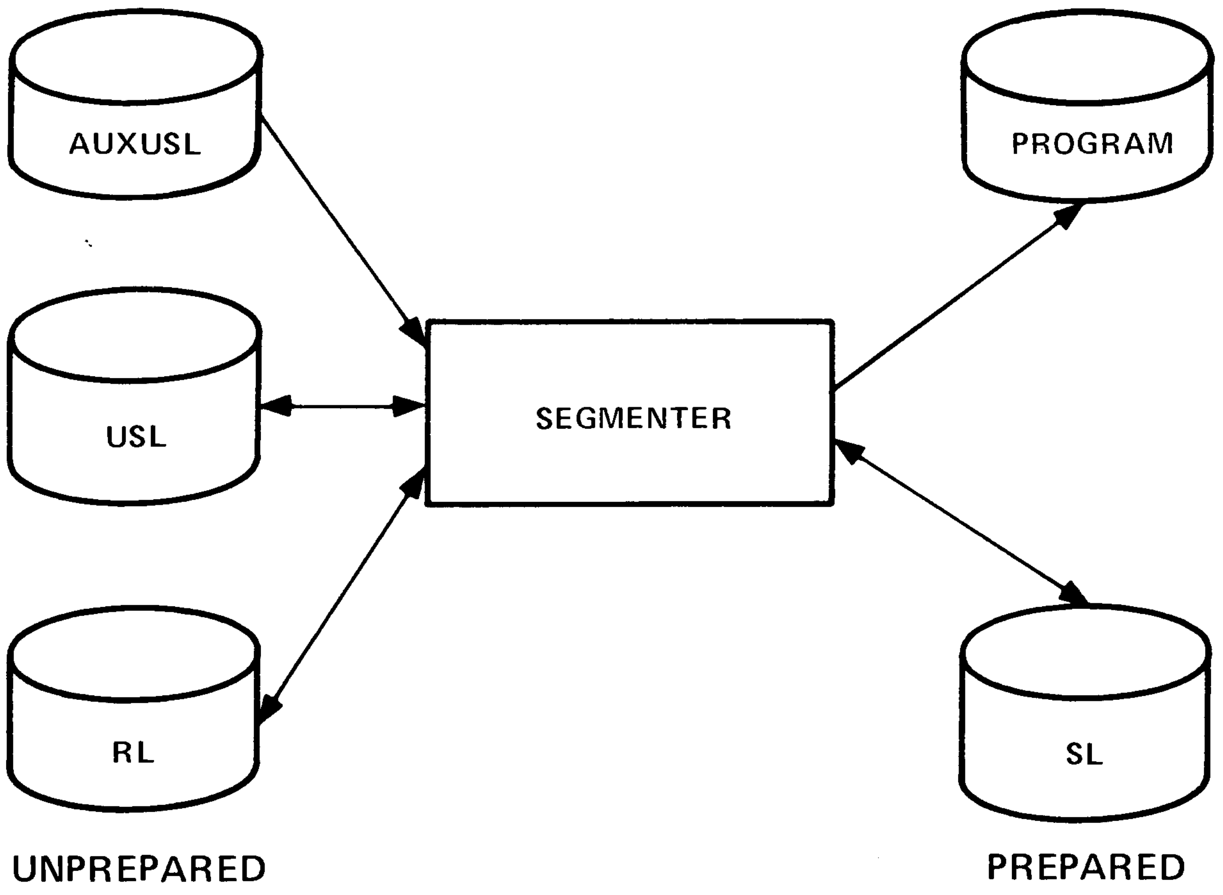
- PREPARES A SEGMENT FROM THE USL (DESIGNATED BY THE USL COMMAND) AND MOVES IT INTO THE SEGMENT LIBRARY DESIGNATED BY THE SL COMMAND.

  - EG.

```
:SEGMENTER
–USL MYUSL
–SL   SL
–ADDSL MYSEG   [,PMAP]
–EXIT
```

IN THE ABOVE SEQUENCE OF COMMANDS THE ADDSL COMMAND INSTRUCTS THE SEGMENTER TO PREPARE MYSEG, A SEGMENT LOCATED IN MYUSL AND PLACE THE PREPARED SEGMENT IN SL.

SEGMENTER (CONT.)



LAB 10

Objective: To create, add to, and use a Segmented Library.

Read the entire lab

1. **Compile the COBOL subprogram into $OLDPASS. The source for the subprogram is in COBSUB1.PUB. It is listed here for reference.**

```
$CONTROL DYNAMIC,MAP,ERRORS=1Ø,SOURCE
******    SUBPROGRAM    *******
 IDENTIFICATION DIVISION.
 PROGRAM-ID. SUBPROG.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 SEVENTY-SEVEN       PIC X(4) VALUE "ZZZZ".
 Ø1 SINGLE-INTEGER      PIC S9(4) COMP VALUE -64.
 Ø1 NUMERIC-DISPLAY     PIC S9(4) VALUE -1234.
 Ø1 PACKED-DECIMAL      PIC S9(4) COMP-3 VALUE -1ØØ.
 Ø1 DOUBLE-INTEGER      PIC S9(6) COMP VALUE -1ØØØØØ.
 Ø1 ASCII-CHARS         PIC A(4) VALUE "ABCD".
 Ø1 FOUR-WORD-INTEGER   PIC S9(12) COMP VALUE -1.
 Ø1 ARRAY.
     Ø5 ITEM            PIC X(2) OCCURS 5 TIMES.
 Ø1 DATE.
     Ø5 MONTH           PIC X(2).
     Ø5 FILLER          PIC X VALUE "/".
     Ø5 DAY             PIC X(2).
     Ø5 FILLER          PIC X VALUE "/".
     Ø5 YEAR            PIC X(2).
 PROCEDURE DIVISION.
 SUBROUTINE SECTION.
 START1.
     MOVE CURRENT-DATE TO DATE.
     DISPLAY "SUB-PROGRAM. DATE = "DATE.
     MOVE SPACES TO ARRAY.
     MOVE MONTH TO ITEM (DOUBLE-INTEGER).

 RETURN1.
     GOBACK.
```

2. **Save $OLDPASS as ZUSL.**

3. **Invoke the SEGMENTER**

   a) **Access ZUSL**
   b) **Build an SL file with 20 records in 1 extent**
   c) **List the SL and verify it is empty**
   d) **List ZUSL and determine the segment names**
   e) **Add the segments from ZUSL to the SL**
   f) **List the SL to check on current items**
   g) **Exit from the Segmenter**

LAB #10 (CONT.)

4. Copy the program file COBPROG1.PUB into your group. (Use FCOPY with NEW option). This is necessary, as system security does not allow an SL to be used that is at a lower level than the program file (i.e. SL at group level and program file at account level). It is listed below for reference.

```
$CONTROL MAP,ERRORS=1Ø,SOURCE
 IDENTIFICATION DIVISION.
 PROGRAM-ID. DEBUGGING.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 SEVENTY-SEVEN        PIC X(4) VALUE "ZZZZ".
 Ø1 SINGLE-INTEGER       PIC S9(4) COMP VALUE 64.
 Ø1 NUMERIC-DISPLAY      PIC S9(4) VALUE 1234.
 Ø1 PACKED-DECIMAL       PIC S9(4) COMP-3 VALUE 1ØØ.
 Ø1 DOUBLE-INTEGER       PIC S9(6) COMP VALUE 1ØØØØØ.
 Ø1 ASCII-CHARS          PIC X(4) VALUE "ABCD".
 Ø1 FOUR-WORD-INTEGER    PIC S9(12) COMP VALUE 1.
 Ø1 ARRAY.
     Ø5 ITEM             PIC X(2) OCCURS 5 TIMES.
 Ø1 DATE.
     Ø5 MONTH            PIC X(2).
     Ø5 FILLER           PIC X VALUE "/".
     Ø5 DAY              PIC X(2).
     Ø5 FILLER           PIC X VALUE "/".
     Ø5 YEAR             PIC X(2).
 PROCEDURE DIVISION.
 MAINØ1 SECTION.
 START1.
     MOVE CURRENT-DATE TO DATE.
     DISPLAY "MAIN PROGRAM. DATE = "DATE.
     MOVE SPACES TO ARRAY.

 CALL-SUBPROGRAM1.
     CALL "SUBPROG".

 TERMINATE1.
     STOP RUN.
```

5. Run the program using the LIB= parameter; request a load map to see that "SUBPROG" was found in the group SL.

6. The program will fail with a BOUNDS VIOLATION. See if you can determine from the program listing, why it failed.

DEL/3000

A SET OF SOFTWARE TOOLS TO EASE THE INTERFACE TO HP 2640/44 TERMINALS.

- ° UTILITY PROGRAM

  DESIGN, MODIFY AND DISPLAY FORMS

- ° CALLABLE PROCEDURES

  ACCESS TERMINAL AS A FILE

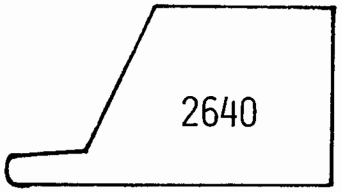
APPLICATIONS FOR DEL/3000

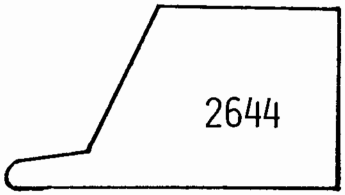
- FINANCE
  - ACCOUNTS PAYABLE/RECEIVABLE
  - GENERAL LEDGER
  - JOURNAL ENTRIES
- MANUFACTURING
  - INVENTORY TRANSACTIONS
  - PURCHASE REQUISITIONS
- ORDER PROCESSING
  - ORDER ENTRY
  - INVOICING
- PERSONNEL
  - EMPLOYEE TRANSACTIONS

... ANY PLACE USER NEEDS FORMS ...

*** NOT INTENDED AS PRODUCTION
KEY-TO-DISC REPLACEMENT ***

ASSUMPTIONS BY DEL/3000





- 4K TERMINAL MEMORY
- ASYNCHRONOUS TERMINAL CONTROLLER (ATC)
- STRAPPED FOR LINE
- NO INTRINSICS PROVIDED FOR 2644 CARTRIDGE MANIPULATION

TERMINOLOGY REVIEW

• PROTECTED FIELD(S)	AREA(S) OF THE SCREEN THAT CANNOT BE ALTERED BY THE TERMINAL USER
• UNPROTECTED FIELD(S)	AREA(S) OF THE SCREEN WHICH ARE CAPABLE OF ACCEPTING DATA FROM THE TERMINAL USER
• BLOCK MODE STRAPPED FOR LINE	WHEN READ IS ISSUED TO TERMINAL, 1 UNPROTECTED FIELD IS RETURNED

FOR FURTHER INFORMATION:

2640 OWNERS MANUAL (02640-90011)

MODUS OPERANDI

1. DESIGN FORM ON PAPER

2. CALL FORMAINT

   - ENTER NEW FORM
   - DEFINE EDIT SPECIFICATIONS
   - DISPLAY FORM

FORMS DESIGNER

3. ACCESS FORM CREATED BY FORMAINT FROM APPLICATION PROGRAM

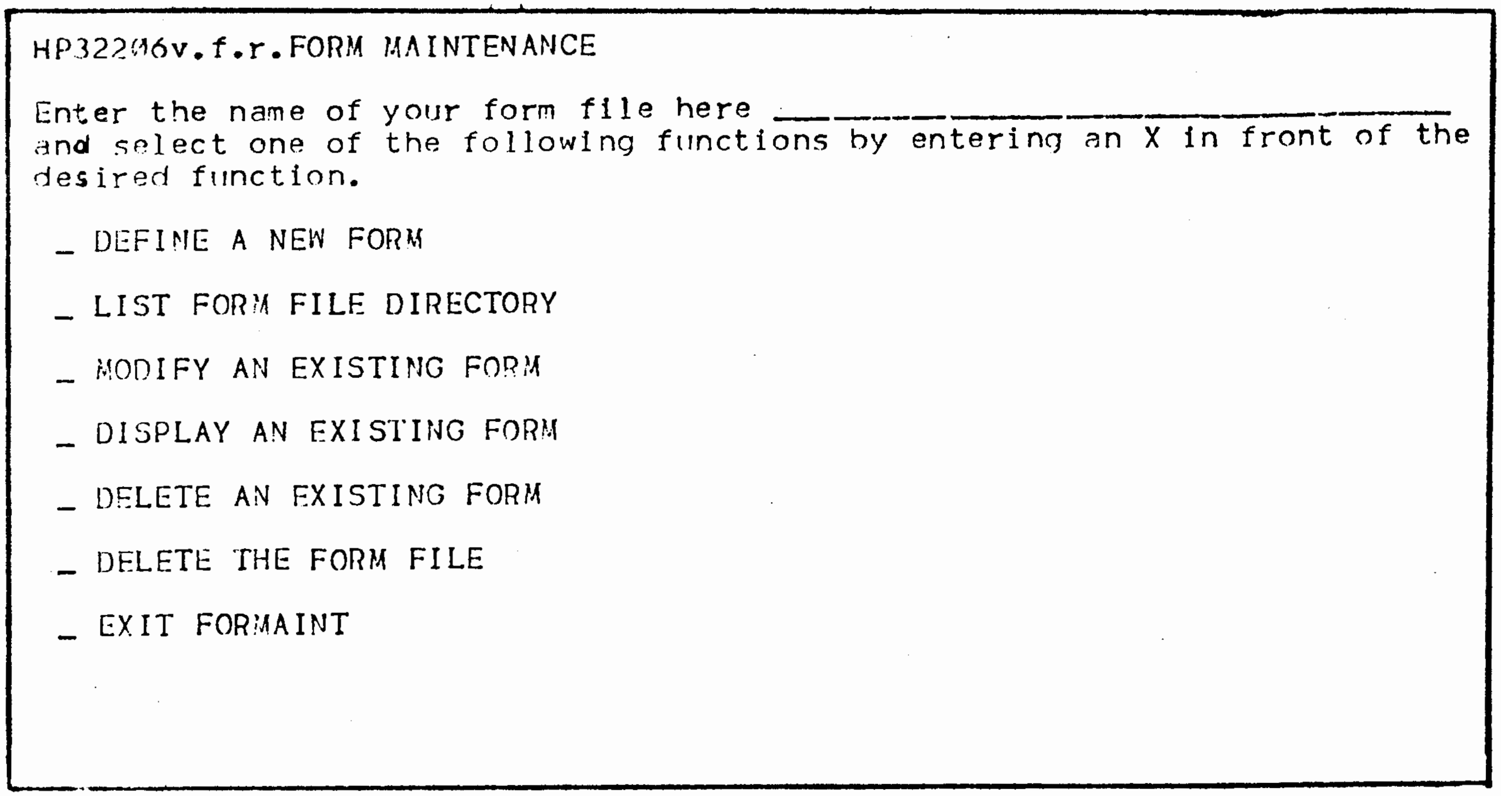
   - DISPLAY FORM
   - ACCEPT/EDIT USER ENTERED DATA
   - CHAIN TO ANOTHER FORM
   - CREATE AN MPE FILE FROM DATA READ BY PROGRAM USING DEL/3000

APPLICATION PROGRAMMER

CREATING FORMS USING DEL/3000

FORMAINT - A UTILITY PROGRAM PROVIDED WITH DEL/3000 TO
CREATE, DISPLAY AND MODIFY FORMS

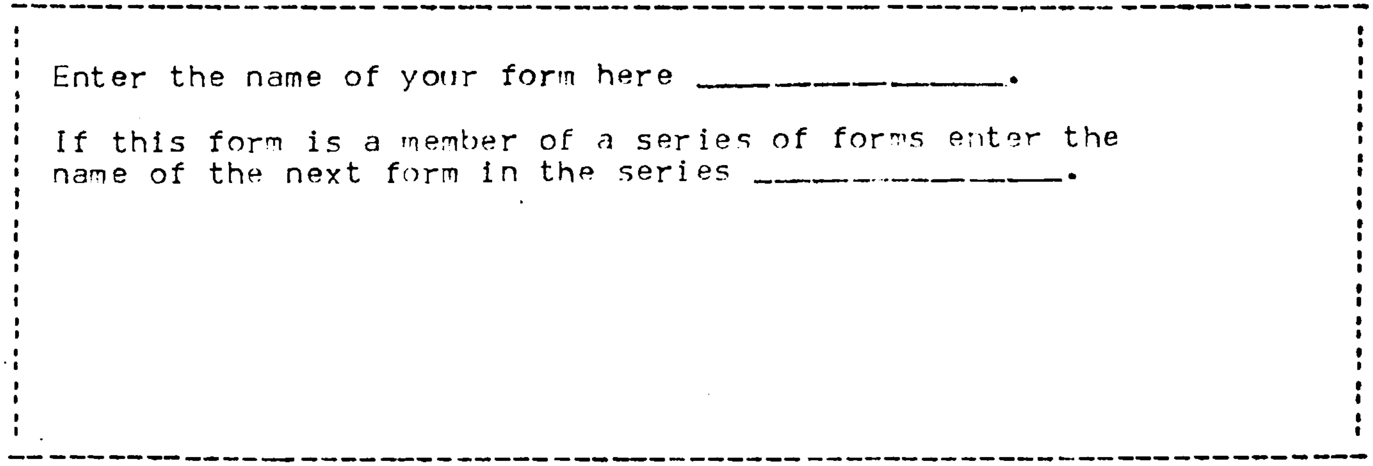
```
: RUN FORMAINT.PUB.SYS
```



```
HP32206v.f.r.FORM MAINTENANCE

Enter the name of your form file here ________________________________
and select one of the following functions by entering an X in front of the
desired function.

 _ DEFINE A NEW FORM

 _ LIST FORM FILE DIRECTORY

 _ MODIFY AN EXISTING FORM

 _ DISPLAY AN EXISTING FORM

 _ DELETE AN EXISTING FORM

 _ DELETE THE FORM FILE

 _ EXIT FORMAINT
```

FORMAINT (CONT'D)

- REQUIRES BLOCK MODE (DEL WILL PROMPT IF KEY NOT DEPRESSED)
- USE "TAB" KEY TO MOVE BETWEEN UNPROTECTED FIELD
- WHEN SCREEN COMPLETE USE "ENTER" KEY TO SEND DATA
- FORM FILE
    - AN ASCII FILE USED/CREATED BY DEL/3000
    - CONFORMS TO MPE STANDARDS
        - FILE NAME MAY BE QUALIFIED
        - PERMANENT
        - RECORD SIZE = 64 BYTES
    - MAY CONTAIN ONE OR MORE FORMS



```
Enter the name of your form here ________________.

If this form is a member of a series of forms enter the
name of the next form in the series ________________.
```

TO SET YOU STRAIGHT!

FORMFILE NAME (35 CHARACTERS-MPE)	FINANCE/MONEY.PUB.INFOSYS
FORM NAME (16 CHARACTERS-DEL)	ACCOUNTSREC ACCOUNTSPAY GENERALLEDGER1 GENERALLEDGER2

FORMAINT (CONT'D)

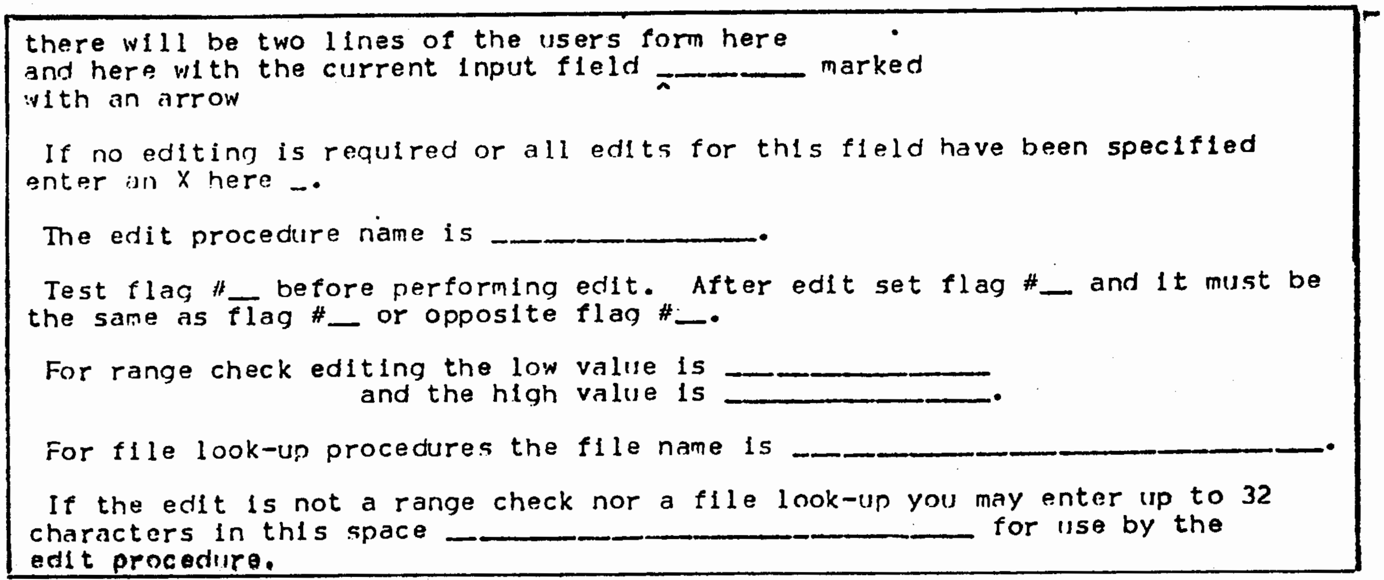
-TO DESIGN YOUR FORM-

- USE "TEST" KEY TO CHECK YOUR TERMINAL'S CAPABILITIES
- DEL/3000 PRESENTS USER WITH BLANK SCREEN
- TO ENTER FORM
  - USE CURSOR POSITIONING
  - DISPLAY ENHANCEMENTS
  - START/END UNPROTECTED FIELD
  - TERMINATE FORM WITH RECORD SEPARATOR (Λ C)
- SEND FORM TO FORMAINT WITH "ENTER"

TURNS ON DISPLAY ENHANCEMENT:

	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Half Bright									X	X	X	X	X	X	X	X
Underline					X	X	X	X					X	X	X	X
Inverse Video			X	X			X	X			X	X			X	X
Blinking		X		X		X		X		X		X		X		X

FORMAINT (CONT'D)



```
there will be two lines of the users form here
and here with the current input field ________ marked
with an arrow                         ^

 If no editing is required or all edits for this field have been specified
enter an X here _.

 The edit procedure name is ________________.

 Test flag #__ before performing edit.  After edit set flag #__ and it must be
the same as flag #__ or opposite flag #__.

 For range check editing the low value is ________________
                         and the high value is ________________.

 For file look-up procedures the file name is ________________________________.

 If the edit is not a range check nor a file look-up you may enter up to 32
characters in this space ________________________________ for use by the
edit procedure.
```

FIELD EDITING

- EDIT PROCEDURE MAY BE

  A. USER WRITTEN

  B. DEL/3000 EDIT PROCEDURE

  1. "ALPHAEDIT" 3. "ANEDIT" 5. "ZEROFILL" 7. "M11CREATE"
  2. "ALPHAFILL" 4. "NUMRCEDIT" 6. "NRANGE" 8. "M11VERIFY"

- EDIT FLAGS

  •• 16 1 BIT FLAGS IN COMMUNICATIONS AREA THAT CAN BE SET AND TESTED BY EDIT PROCEDURES.

  •• HELPS WITH LOGICAL EDITS BETWEEN FIELDS.

OTHER FUNCTIONS OF FORMAINT

- MODIFY FORM

Enter the name of the form to be modified____________________.

If the form to be modified is a member of a series of forms and the name of the next form in the series is to be changed enter the name of the next form in the series ____________________.

- DISPLAY FORM

Enter the name of the form to be displayed ____________________.

If you want the edit specifications displayed enter the name of the destination file ____________________.

- ANY OUTPUT FILE CAN BE SPECIFIED
  - DISC
  - OTHER — USE BACKREFERENCE (*NAME) TO PREVIOUS FILE COMMAND
  - OUTPUT FORMAT IS 80 BYTE, ASCII
- IF DISPLAY ON TERMINAL ONLY FORM IS SHOWN
- IF DISPLAY ON OTHER DEVICE DEL/3000 WILL OUTPUT
  - FORM DESCRIPTIVE INFORMATION
  - FORM
  - INPUT EDIT SPECIFICATIONS

FORM DESCRIPTIVE INFORMATION

```
    FORM NAME IS PCASH                        CREATED   3/25/76  14:03
THIS FORM CONTAINS 9 INPUT FIELDS. TOTAL INPUT LENGTH IS 64 BYTES.
THERE ARE 8 EDITS SPECIFIED.
```

FORM

```
                 PETTY CASH REQUEST

TO:                                 DATE:   /  /

DESC:                               ACCT:       LOC:       AMOUNT:    .
```

INPUT EDIT SPECIFICATIONS

```
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   3    8      1       20        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ANEDIT                   0    0    0      0
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   3   45     21        2        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                   0    0    0      0     0000000000000001000000000000012
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   3   48     23        2        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                   0    0    0      0     0000000000000001000000000000031
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   3   51     25        2        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                   0    0    0      0     0000000000000076000000000000076
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   6   10     27       26        0
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   6   45     53        4        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ZEROFILL                 0    0    0      0
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   6   57     57        4        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
ZEROFILL                 0    0    0      0
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   6   71     61        2        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                   0    0    0      0     0000000000000000000000000000049
  FIELD LOCATION     FIELD     NUMBER
 ROW  COL IN INPUT  LENGTH   OF EDITS
   6   74     63        2        1
PROCEDURE NAME   FLAGS TEST SET SAME OPPOSITE PROCEDURE DATA
NRANGE                   0    0    0      0     0000000000000000000000000000099
```

Computer Museum

OTHER FUNCTIONS OF FORMAINT (CONT'D)

- DELETE AN EXISTING FORM

```
Enter the name of the form to be deleted ____________________ .
```

- PURGE AN ENTIRE FORM FILE

```
Is form file 'filename' to be deleted?
Enter YES or NO _____
```

- EXIT FORMAINT

***AT ANY TIME YOU MAY RETURN TO FUNCTION "MENU" IN FORMAINT

- 2640 – HOLD CNTRL AND PRESS F8 KEY
- 2644 – DEPRESS F8 KEY

LAB EXERCISE (DEL/3000)

Following you will find a layout for your DEL/3000 exercise.

Before you begin to enter the form press the "TEST" button on your terminal to check your terminal's capabilities. You should use some display enhancement, which one depends on your terminal capabilities and personal preference.

The fields on the form do not have to start in any particular place but the "TO" field is first, month next, etc. The field lengths <u>are</u> significant.

Enter "FORMAINT" to create the screen and enter the edit specifications. If you have to modify form with "MODIFY FORM" you will have to re-enter all your edit specifications.

In the next lab you will be modifying a program to access this form you are entering through FORMAINT.

WORK EXERCISE

PETTY CASH REQUEST

TO: □ DATE: □/□/□

DESC: □ ACCT: □ LOC: □ AMOUNT: □

EDIT SPECIFICATIONS FOR WORK EXERCISE
(no edit flags needed)

FIELD	LENGTH	TYPE OF EDIT	
TO	20	ALPHANUMERIC	
MONTH	2	RANGE	01-12
DAY	2	RANGE	01-31
YR	2	RANGE	76
DESC	26	NO EDIT	
ACCT	4	ZEROFILL	
LOCATION	4	ZEROFILL	
AMOUNT			
DOLLARS	2	RANGE	00-49
CENTS	2	RANGE	00-99

USER CALLABLE PROCEDURES

- ACCESSIBLE FROM COBOL, FORTRAN, SPL, BASIC
- PREFIX NAME WITH "C" FOR COBOL
- COMMUNICATIONS AREA REQUIRED
  - 128 WORDS
  - FIRST WORD FOR STATUS (COMP-COBOL
    INTEGER-OTHERS)
- FOUR TYPES OF CALLABLE PROCEDURES
  - FORMS ACCESS
  - TERMINAL ACCESS
  - INPUT EDIT
  - HIGH LEVEL

SUMMARY OF CALLABLE PROCEDURES

OPENFORM	(COMMAREA,FORMFILE)
FINDFORM	(COMMAREA,FORMNAME,LENGTH,NEXTFORM)
GETFORM	(COMMAREA,BUFFER,LENGTH)
NEXTEDIT	(COMMAREA,BUFFER)
CLOSEFORM	(COMMAREA)
OPENTERM	(COMMAREA,TERMNAME)
WRITETERM	(COMMAREA,BUFFER,LENGTH)
READTERM	(COMMAREA,BUFFER,LENGTH)
TERMSTATUS	(COMMAREA,BUFFER)
CLOSETERM	(COMMAREA)
ALPHAEDIT	(COMMAREA,EDITSPEC,BUFFER)
ALPHAFILL	(COMMAREA,EDITSPEC,BUFFER)
ANEDIT	(COMMAREA,EDITSPEC,BUFFER)
NUMRCEDIT	(COMMAREA,EDITSPEC,BUFFER)
ZEROFILL	(COMMAREA,EDITSPEC,BUFFER)
NRANGE	(COMMAREA,EDITSPEC,BUFFER)
M11CREATE	(COMMAREA,EDITSPEC,BUFFER)
M11VERIFY	(COMMAREA,EDITSPEC,BUFFER)
SHOWFORM	(COMMAREA,FORMNAME,NEXTFORM,BUFFER,LENGTH)
EDITFIELD	(COMMAREA,EDITSPEC,BUFFER)

SUMMARY OF PARAMETERS

COMMAREA	DEL/3000 COMMUNICATIONS AREA 128 WORDS OF CONTIGUOUS STORAGE USED BY DEL/3000 FOR GLOBAL STORAGE AREA FIRST WORD FOR USER (RETURNED STATUS) OTHER 127 WORDS FOR USE BY DEL/3000
FORMFILE	CHARACTER DATA ITEM MAXIMUM LENGTH 35 CHARACTERS CONTAINS NAME OF FORMFILE
FORMNAME	CHARACTER DATA ITEM 16 CHARACTERS CONTAINS NAME OF FORM
NEXTFORM	CHARACTER DATA ITEM 16 CHARACTERS CONTAINS NAME OF NEXT FORM
LENGTH	NUMERIC DATA ITEM 1 WORD USED FOR PASSING BUFFER LENGTH TO DEL/3000 USED BY DEL/3000 TO RETURN DATA LENGTH
BUFFER	CHARACTER DATA ITEM MINIMUM LENGTH 64 BYTES CONTIGUOUS STORAGE AREA USED BY DEL/3000 TO RETURN DATA AND AREA ALSO USED TO PASS DATA TO DEL/3000
TERMNAME	CHARACTER DATA ITEM MAXIMUM LENGTH 8 CHARACTERS FIELD WILL BE USED AS FORMALDESIGNATOR FOR TERMINAL FILE
EDITSPEC	CHARACTER DATA ITEM 72 CHARACTERS CONTAINS EDIT SPECIFICATIONS CREATED BY FORMAINT

IN YOUR APPLICATION PROGRAM

```
          .                                    .
          .                                    .
          .                                    .
      "OPENFORM"                           "OPENFORM"
      (CHECK STATUS)                       (CHECK STATUS)
      "OPENTERM"                           "OPENTERM"
      (CHECK STATUS)                       (CHECK STATUS)
          .                                    .
          .                                    .
          .                                    .
(A)   "FINDFORM"
      (CHECK STATUS)

      "GETFORM"
      (CHECK STATUS)                       "SHOWFORM"
                                           (CHECK STATUS)
(B)   "WRITETERM"
      (CHECK STATUS)

      "READTERM"
      (CHECK STATUS)                           .
          .                                    .
          .                                    .
          .
      "NEXTEDIT"
      (CHECK STATUS)                       "EDITFIELD"
      "NUMRCEDIT"                          (CHECK STATUS)
      (CHECK STATUS)
          .
          .
          .
      "NEXTEDIT"
      (CHECK STATUS)                       "EDITFIELD"
      "ALPHAEDIT"                          (CHECK STATUS)
      (CHECK STATUS)
          .
          .
          .
      SAME FORM?  GO TO (B)                    .
                                               .
      NEW FORM?   GO TO (A)                    .

      "CLOSEFORM"                          "CLOSEFORM"
      (CHECK STATUS)                       (CHECK STATUS)
      "CLOSETERM"                          "CLOSETERM"
      (CHECK STATUS)                       (CHECK STATUS)
          .                                    .
          .                                    .
          .                                    .
```

FORMS ACCESS PROCEDURES

TO OPEN A FORM FILE

- "OPENFORM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- NAME OF YOUR FORM FILE

STATUS RETURNED

- 0 OPERATION SUCCESSFUL
- -1 NOT A FORM FILE
- >0 ERROR CODE FROM "FCHECK"

FORMS ACCESS PROCEDURES (CONT'D)

TO LOCATE A FORM IN A FORM FILE

- "FINDFORM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- NAME OF YOUR FORM
- LENGTH OF YOUR FORM (RETURNED PARAMETER)
- NAME OF NEXT FORM (RETURNED PARAMETER)

STATUS RETURNED

- 0 OPERATION SUCCESSFUL
- -1 FORM NOT IN FORM FILE
- >0 ERROR CODE FROM "FCHECK"

NOTE: OPENFORM MUST HAVE BEEN CALLED

FORMS ACCESS PROCEDURES (CONT')

TO PREPARE/MOVE FORM TO USER BUFFER

- "GETFORM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- USER BUFFER (RETURNED PARAMETER)
- LENGTH OF USER BUFFER (PASSED AND RETURNED PARAMETER)

STATUS RETURNED

- 0 OPERATION SUCCESSFUL
- >0 ERROR CODE FROM "FCHECK"

NOTE: FINDFORM MUST HAVE ALREADY BEEN CALLED

FORMS ACCESS PROCEDURES (CONT'D)

TO GET NEXT EDIT SPECIFICATION OR TO POSITION FOR EDITING

- "NEXTEDIT"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- USER BUFFER (RETURNED PARAMETER)

RETURNED STATUS

• 0	OPERATION SUCCESSFUL
• -1	LAST EDIT SPECIFICATION HAS ALREADY BEEN ACCESSED
• >0	ERROR CODE FROM "FCHECK"

NOTE: FINDFORM MUST HAVE BEEN CALLED

FORMS ACCESS PROCEDURES (CONT'D)

TO CLOSE CURRENT FORM FILE

- "CLOSEFORM"

REQUIRED PARAMETER

- DEL/3000 COMMUNICATIONS AREA

RETURNED STATUS

- 0 OPERATION SUCCESSFUL
- >0 ERROR CODE FROM "FCHECK"

TERMINAL ACCESS PROCEDURES

TO OPEN AND VERIFY TERMINAL FILE

- "OPENTERM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- TERMINAL NAME

RETURNED STATUS

- 0 OPERATION SUCCESSFUL
- -1 TERMINAL NOT A 2640
- >0 CANNOT OPEN ("FCHECK" ERROR CODE)

NOTE: IF LINE MODE (ATC) COMMUNICATIONS AREA WILL BE SET TO EMULATE PAGE MODE

TERMINAL ACCESS PROCEDURES (CONT'D)

WRITE TO (DISPLAY ON) TERMINAL

- "WRITETERM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- USER BUFFER
- LENGTH OF USER BUFFER

RETURNED STATUS

- 0   OPERATION SUCCESSFUL
- >0   ERROR CODE FROM "FCHECK"

NOTE: MUST HAVE BEEN PRECEDED BY A CALL TO OPENTERM

TERMINAL ACCESS PROCEDURES (CONT'D)

READ DATA FROM TERMINAL

- "READTERM"

REQUIRED PARAMETERS

- DEL/3000 COMMUNICATIONS AREA
- USER BUFFER
- LENGTH (RETURNED PARAMETER)

RETURNED STATUS

- 0 OPERATION SUCCESSFUL
- >0 ERROR CODE FROM "FCHECK"

NOTE: MUST BE PRECEDED BY A CALL TO OPENTERM

TERMINAL ACCESS PROCEDURES (CONT'D)

GET TERMINAL STATUS

- "TERMSTATUS"

REQUIRED PARAMETER

- DEL/3000 COMMUNICATIONS AREA
- USER BUFFER   (RETURNED PARAMETER)

STATUS RETURNED

- 0   OPERATION SUCCESSFUL
- >0   ERROR CODE FROM "FCHECK"

NOTE: OPENTERM MUST HAVE ALREADY BEEN CALLED

TERMINAL ACCESS PROCEDURES (CONT'D)

TO CLOSE TERMINAL FILE

- "CLOSETERM"

REQUIRED PARAMETER

- DEL/3000 COMMUNICATIONS AREA

RETURNED STATUS

- 0 OPERATION SUCCESSFUL
- >0 ERROR CODE FROM "FCHECK"

INPUT EDIT PROCEDURES

- CALLED BY APPLICATION PROGRAM
- RETURNS PASS/FAIL INDICATION $\begin{bmatrix} 0 = \text{PASS} \\ -1 = \text{FAIL} \end{bmatrix}$
- DOES NOT INTERACT TO CORRECT ERRONEOUS DATA - USER CAN DO OWN DATA CORRECTION
- EACH CALL MUST BE PRECEDED BY "NEXTEDIT" TO POSITION FOR EDITING
- ALL EDIT PROCEDURES USE SAME PARAMETERS
    - DEL/3000 COMMUNICATIONS AREA
    - EDIT SPECIFICATIONS AS RETURNED FROM "NEXTEDIT"
    - USER BUFFER CONTAINING DATA TO BE EDITED

EDIT PROCEDURES (CONT'D)

"ALPHAEDIT"   (COMMAREA,EDITSPEC,BUFFER)

- MUST CONTAIN LETTERS A-Z
- NO SPACES

"ALPHAFILL"   (COMMAREA,EDITSPEC,BUFFER)

- LETTERS A-Z
- SPACES ALLOWED TO RIGHT OF LAST ALPHA CHARACTER

"ANEDIT"   (COMMAREA,EDITSPEC,BUFFER)

- ALPHANUMERIC
- LETTERS A-Z
- SPACE
- DIGIT 0-9
- NO SPECIAL CHARACTERS

"NUMRCEDIT"   (COMMAREA,EDITSPEC,BUFFER)

- NUMBERIC
- DIGITS 0-9
- UNSIGNED DATA ONLY

"ZEROFILL"   (COMMAREA,EDITSPEC,BUFFER)

- NUMERIC
- ZERO FILL TO LEFT OF DIGITS ENTERED

"NRANGE"   (COMMAREA,EDITSPEC,BUFFER)

- PROCEDURE CALLS "ZEROFILL"
- COMPARE DATA INPUT AGAINST HIGH AND LOW RANGE ENTERED WITH EDIT DEFINITIONS

EDIT PROCEDURES (CONT'D)

"M11CREATE" (COMMAREA,EDITSPEC,BUFFER)

- PROCEDURE CALLS "ZEROFILL"
- WILL GENERATE MODULO ELEVEN CHECK DIGIT
- PROCEDURE WILL INSERT THE CHECK DIGIT AS RIGHTMOST DIGIT OF INPUT FIELD

"M11VERIFY" (COMMAREA,EDITSPEC,BUFFER)

- PROCEDURE CALLS "ZEROFILL"
- GENERATES CHECK DIGIT AND COMPARES TO RIGHTMOST DIGIT IN INPUT FIELD.

HIGH LEVEL INTERFACE PROCEDURE

TO DISPLAY FORM AND READ INPUT

- "SHOWFORM"

  USES "FINDFORM" TO LOCATE FORM DEFINITION

  USES "GETFORM" TO ACCESS FORM DEFINITION

  USES "WRITETERM" TO SEND FORM TO TERMINAL

  USES "READTERM" TO READ INPUT DATA

- REQUIRED PARAMETERS

  - DEL/3000 COMMUNICATIONS AREA
  - NAME OF YOUR FORM
  - NAME OF THE NEXT FORM
  - USER BUFFER (PASSED AND RETURNED)
  - LENGTH OF BUFFER (PASSED AND RETURNED)

- STATUS RETURNED

0	OPERATION SUCCESSFUL
-1	FORM CANNOT BE LOCATED
>0 ≤999	ERROR CODE FORM "FCHECK" (WRITETERM OR READTERM)
>1000	ERROR CODE FROM "FCHECK" + 1000 (FINDFORM OR GETFORM)

HIGH LEVEL INTERFACE PROCEDURE

TO EDIT NEXT INPUT FIELD

- "EDITFIELD "
  - PRESUMES FINDFORM AND READTERM HAVE BEEN CALLED
  - USES NEXTEDIT TO GET EDIT SPECIFICATION
  - CALLS APPROPRIATE EDIT PROCEDURE FOR REQUIRED EDITING
    - ANEDIT
    - ZEROFILL
    - ETC
  - MAY NOT BE USED TO CALL USER SUPPLIED EDIT PROCEDURES
- REQUIRED PARAMETERS
  - DEL/3000 COMMUNICATIONS AREA
  - BUFFER FOR EDIT SPECIFICATIONS (RETURNED)
  - BUFFER CONTAINING DATA ITEM TO BE EDITED (FROM READTERM)
- STATUS RETURNED

0	OPERATION SUCCESSFUL
-3	REQUIRED EDIT NOT ONE OF DEL/3000 PROCEDURES
-2	LAST EDIT SPECIFICATION HAS BEEN ACCESSED
-1	FAILED EDIT CHECK
>0	ERROR CODE FROM "FCHECK"

LAB #11 DEL/3000

OBJECTIVE: TO GIVE AN INTRODUCTION TO DEL/3000 CALLABLE PROCEDURES

TASK: Using FCOPY, copy the source for the lab into your group.

```
a.  DELCLAB.PUB     (COBOL)
        or
b.  DELFLAB.PUB     (FORTRAN)
```

Using the EDITOR modify the source to include the appropriate "CALL" statements as indicated by the comments in the program. The lines with #'s across are where the changes should be made. An offline listing of the program should help you to make the changes.

The program writes the valid output records to a session temporary disc file. The invalid records are written to a line printer file with the fields in error indicated by a line of "*" underneath the field in error.

Compile, prep, and run the program.

> NOTE: if you list the program source to your terminal, turn on DISPLAY FUNCTIONS before the output starts. The escape codes will be listed instead of executed.

THANK YOU FOR ATTENDING

HP 3000: A COMPREHENSIVE INTRODUCTION

GENERAL SYSTEMS DIVISION TRAINING CENTER
WISHES YOU GREAT SUCCESS
WITH YOUR HP 3000.