

DOS

HANDOUTS

PDP-11 DISK OPERATING SYSTEM SOFTWARE (DOS)

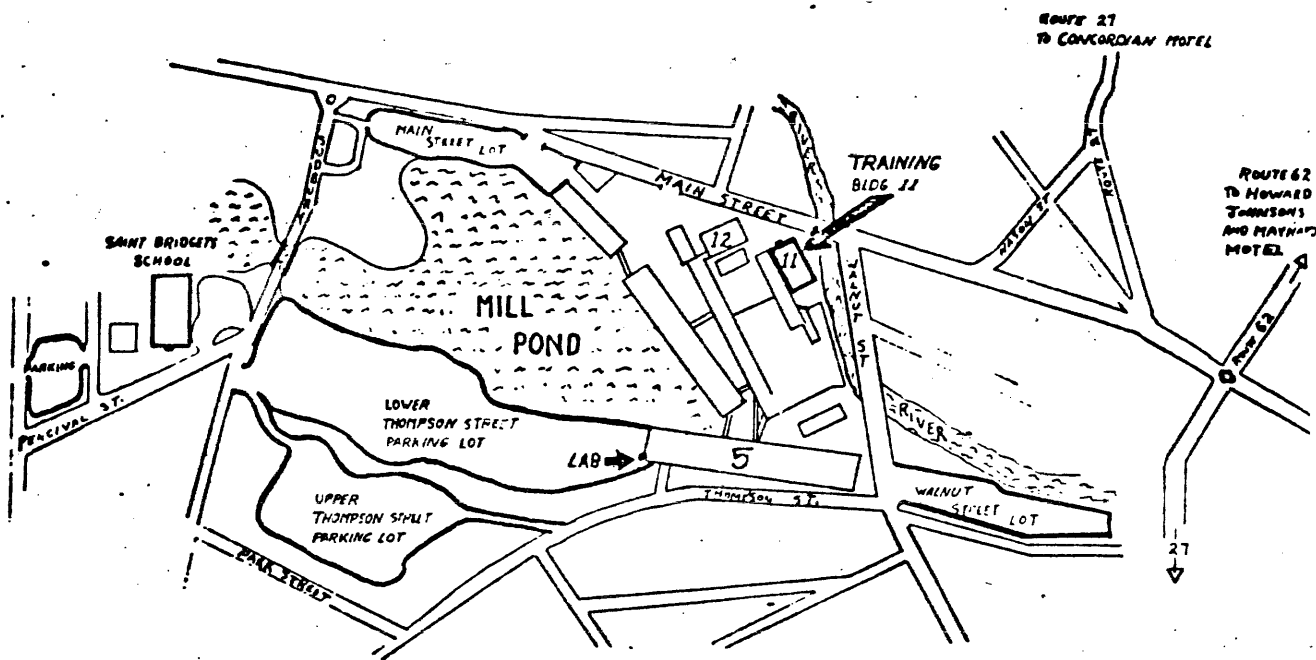
This course is designed to teach the student how to:

1. Generate a Disk Operating System from DECtape or paper tape.
2. Write, run, and modify programs using programmed monitor requests and the DOS software programs.
3. Communicate with the system by means of keyboard commands.

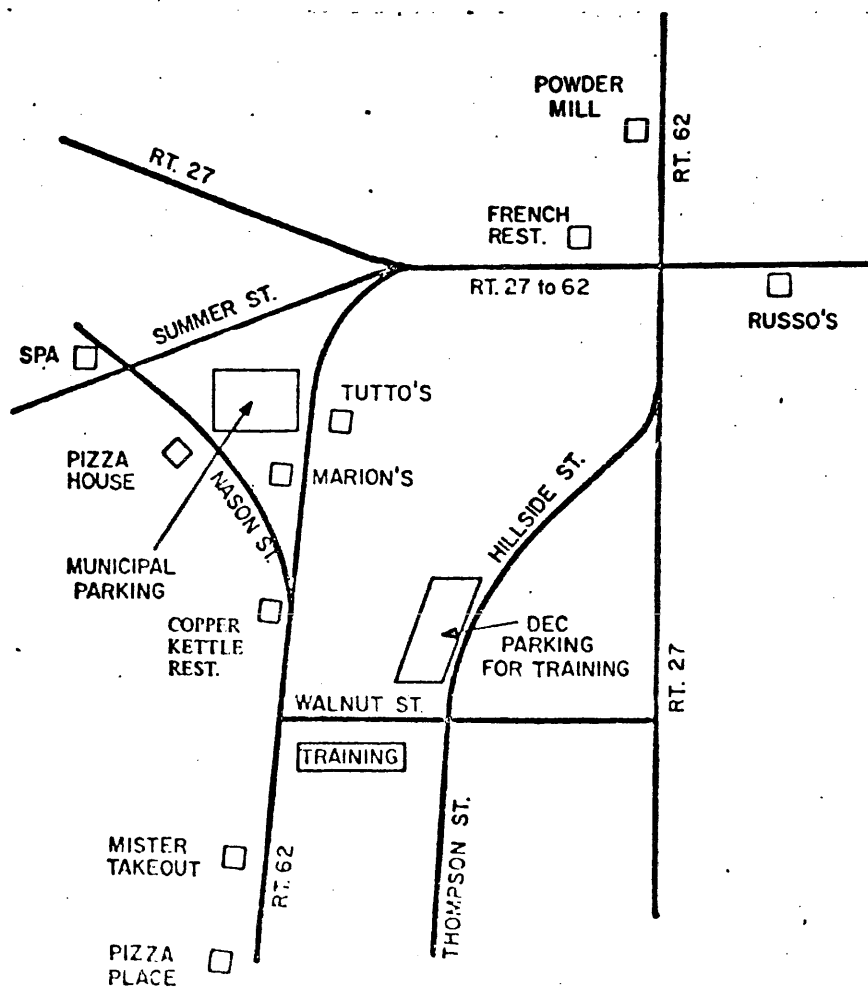
LENGTH: 5 Days

PREREQUISITES: The student must be thoroughly familiar with the PDP-11 instruction set and programming techniques, and the following PDP-11 paper tape software programs: Assembler (PAL-11), Text Editor (ED-11), On-line Debugging Technique (ODT-11), Extended On-line Debugging Technique (ODT-11X), Core Memory Dumps (DUMPTT) (DUMPAB), Input-Output Executive (IOX), Floating Point Math Package (FPP-11). Formal training in these areas can be obtained by attending any of the following courses: PDP-11 PAPER TAPE SOFTWARE (STANDARD), PDP-11 PAPER TAPE SOFTWARE (ACCELERATED), or PROGRAMMING THE PDP-11.

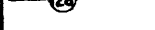
CONTENT: This is designed to be a *user* course, and does not present a detailed examination of the DOS monitor. It contains an overview of the monitor program and file structure, and covers the use of programmed monitor requests, keyboard commands, and the following DOS software programs: BATCH-11, MACRO-11 Assembler, Text Editor (EDIT-11), Relocatable On-line Debugging Technique (ODT-11R), Linker (LINK-11), Librarian (LIBR-11), File Utility Program (PIP), and system generation (CILUS, SYSLOD, ROLLIN). A portion of the course time is allotted to supervised laboratory sessions.



DIGITAL EQUIPMENT CORPORATION TRAINING



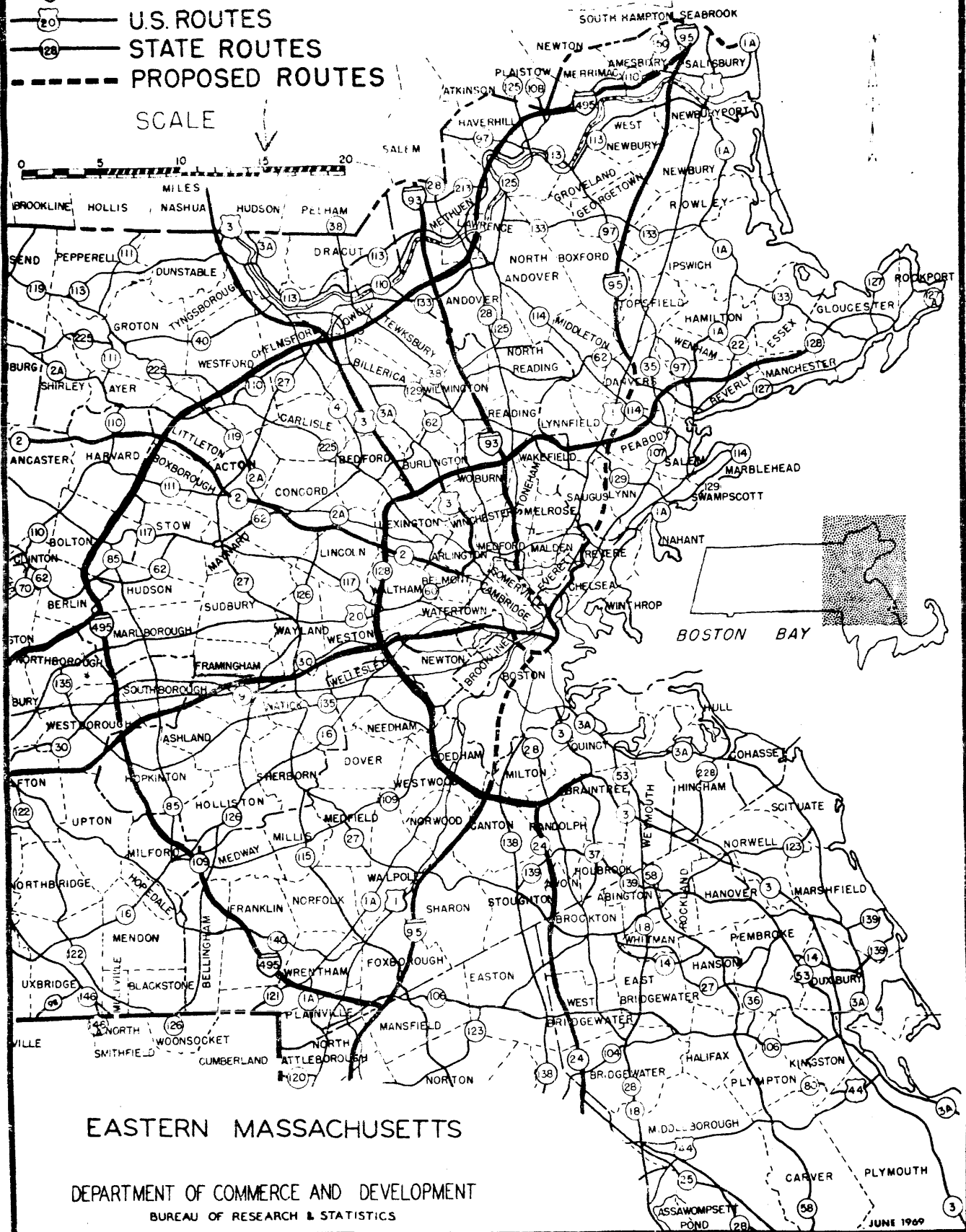
LEGEND

-  INTERSTATE
-  U.S. ROUTES
-  STATE ROUTES
-  PROPOSED ROUTES

SCALE



NEW ENGLAND AREA CONSULTANTS, INC.



EASTERN MASSACHUSETTS

DEPARTMENT OF COMMERCE AND DEVELOPMENT
BUREAU OF RESEARCH & STATISTICS

MONDAY	I. INTROUDCTION II. DOS OVERVIEW W/FILE LAYOUTS III. INPUT/OUTPUT PROCESSING OVERVIEW
TUESDAY	I. MACRO-11 II. TRAN LEVEL PROCESSING III. CREATION OF A CONTIGUOUS FILE
WEDNESDAY	I. BLOCK LEVEL PROCESSING II. CREATION OF A LINKED FILE III. RECORD LEVEL PROCESSING IV. EDITOR AND PIP OPERATIONS V. LAB
THURSDAY	I. READ/WRITE PROCESSING II. PROGRAMMED MONITOR REQUESTS III. SYSTEM SOFTWARE-LINKER, LIBR, ODT, FILDMP IV. OVERLAY BUILDING AND EXECUTION V. LAB
FRIDAY	I. SYSGEN W/ROLLIN II. BATCH III. CILUS IV. LAB (SYSTEM GENERATION)

LOCATION	CONTENT	CONTENT SYMBOL
40	START ADDRESS OF THE SYSTEM VECTOR TABLE.	SVT.
42	BASE ADDRESS FOR THE GENERAL SYSTEM EXIT.	S.XIT
44	START ADDRESS FOR THE REGISTER-SAVE SUBROUTINE	S.RSAV
46	START ADDRESS FOR THE REGISTER-RESTORE SUBRTN.	S.RRES
50	START ADDRESS FOR THE DRIVER QUEUE SUBROUTINE	S.CDB
52	START ADDRESS FOR THE DRIVER DEQUEUE SUBRTN.	S.CDQ
54	START ADDRESS FOR THE GET-BUFFER SUBROUTINE	S.GTB
56	START ADDRESS FOR THE RELEASE-BUFFER SUBRTN.	S.RLB

USAGE OF THE FIXED VECTOR LOCATIONS.

Starts at 2000 location ?

SVT:			
EOM:	.WORD	Ø	; END OF PERMANENT MONITOR
TOB:	.WORD		; END OF BUFFER AREA
CSA:	.WORD	37776	; LAST ADDRESS IN AVAILABLE CORE
PLA:	.WORD	Ø	; PROGRAM LOAD ADDRESS
SCW:	.WORD	Ø	; SYSTEM CONFIGURATION INDICATORS
			; BIT Ø CLOCK CYCLE INDICATOR, USED BY C
			; 6Ø CLCLE = Ø, 5Ø CYCLE = 1
			; BIT 6 CLOCK TYPE INDICATOR
			; KW11-L = Ø, KW11-P = 1
			; BIT 7 NO CLOCK. INDICATOR
			; NO CLOCK = Ø, CLOCK = 1
			; BIT 13 SYSTEM LOAD BIT
			; LOAD = 1 OTHERWISE = Ø
			; BIT 14 PRIVILEGED USER BIT
			; BIT 15 AC INDICATOR, USED BY BATCH SYSTEM
BAT:	.WORD	Ø	; POINTS TO DYNAMIC ASSIGNMENT TABLE
DCO:	.WORD	Ø	; CHAIN OF INITIED DDBS
MUS:	.BYTE	Ø, Ø	; MONITOR USER SWITCH ←
OSW:	.WORD	Ø	; OTHER SWITCH
PSA:	.WORD	Ø	; PROGRAM START ADDRESS
DSA:	.WORD	Ø	; DDT STARTING ADDRESS
RSA:	.WORD	Ø	; RESTART ADDRESS - RESTART - LOADS this address
WRA:	.WORD	WTL	; WAIT RETURN ADDRESS
DAT:	.WORD	Ø	; DATE IN JULIAN - 7Ø, ØØØ
TOD:	.WORD	Ø, Ø	; TIME OF DAY CLOCK CELLS - TICKS
UIC:	.WORD	Ø	; USER IDENTIFICATION CODE
PGN:	.WORD	Ø, Ø	; PROGRAM NAME (MOD 4Ø)
MRT:	.WORD	MRT.	; START OF RESIDENCY TABLE
DDL:	.WORD	Ø	; START OF DEVICE LIST
SSP:	.WORD	Ø	; STACK POINTER SAVE (DUMP)
BFS:	.WORD	Ø	; START OF BUFFER ALLOC. TABLE
BFE:	.WORD	Ø	; END OF BUFFER ALLOC TABLE
WTL:	BR	.	; WAIT LOOP FOR WAITS
KBA:	.WCRO	Ø	; POINTER TO KB IN DDL
MSB:	.WORD	MSB.	; POINTER TO MAIN SWAP AREA
STR:	.WORD	Ø	; STACK BASE (BASE + 2)
DFU:	.BYTE	2, 1	; SCRATCH UIC
KBP:	.WORD	KBPTR	; KEYBOARD POINTER
CIL:	.WORD	Ø	; CIL BASE
KSP:	.WORD	KBI.-2	; POINTER TO KEYBOARD SWAP BUFFER
BKZ:	.WORD	Ø	; DISK BLOCK SIZE INDICATOR
			; 4 = 64 WORD BLOCKS, 2 = 256 WORD BLOCKS
M.IOMX:	.BYTE	Ø	; MAXIMUM I/O EMT
M.EMAX:	.BYTE	Ø	; MAXIMUM EMT IPERMITTED
TPNPRT:	.WORD	Ø	; ABSOLUTE DISK ADDRESS OF MRT COPY USED BY TMOI
MUOTBL:	.WORD	Ø	; FOR MUO - Multi USER Dos

Low Byte 1 - Program loaded -1 Program Stopped	High Byte 1 Running -1 Waiting
--	--------------------------------------

System Vector TABLE (Address location 40)

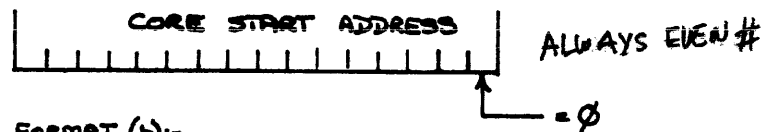
MONITOR RESIDENCY TABLE

MRT:

0	
1	
2	
3	
4	
5	
6	
7	
10	
11	
12	
13	REQUESTS FOR I/O SERVICES
14	
15	
16	
17	
20	
21	
22	
23	
24	
25	
26	
27	
30	'EMERGENCY' SERVICES USING THE KEYBOARD SWAP BUFFER
31	
32	
33	
34	
35	
36	
37	
40	OTHER PROGRAM SERVICES
41	
42	
43	
44	
45	

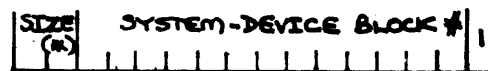
FORMAT (a):-

CORRESPONDING ROUTINE IS IN CORE:-



FORMAT (b):-

CORRESPONDING ROUTINE IS IN THE
SYSTEM LIBRARY EXTERNALLY:-



FORMAT (c):-

NO CORRESPONDING ROUTINE EXISTS:-



(N) = # OF 64-WORD BLOCKS
(00 = 4)

Figure 2-3: MONITOR RESIDENCY
TABLE FORMAT.

Set up when system is built

DDL.:

—	ALWAYS	—
—	ENTRY	—
—	FOR	—
—	SYSTEM	—
—	DEVICE	—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#2	—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#3	—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#4	—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#5	—
—		—
—		—
—		—
—		—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#n-1	—
—	ENTRY	—
—	FOR	—
—	DEVICE	—
—	#n	—

DDL.END:

DEVICE NAME (Radix-50)
MEMORY START ADDRESS
INTERRUPT VECTOR ADDRESS
DDL END ADDRESS ADDR OF LAST ENTRY INTO DDL

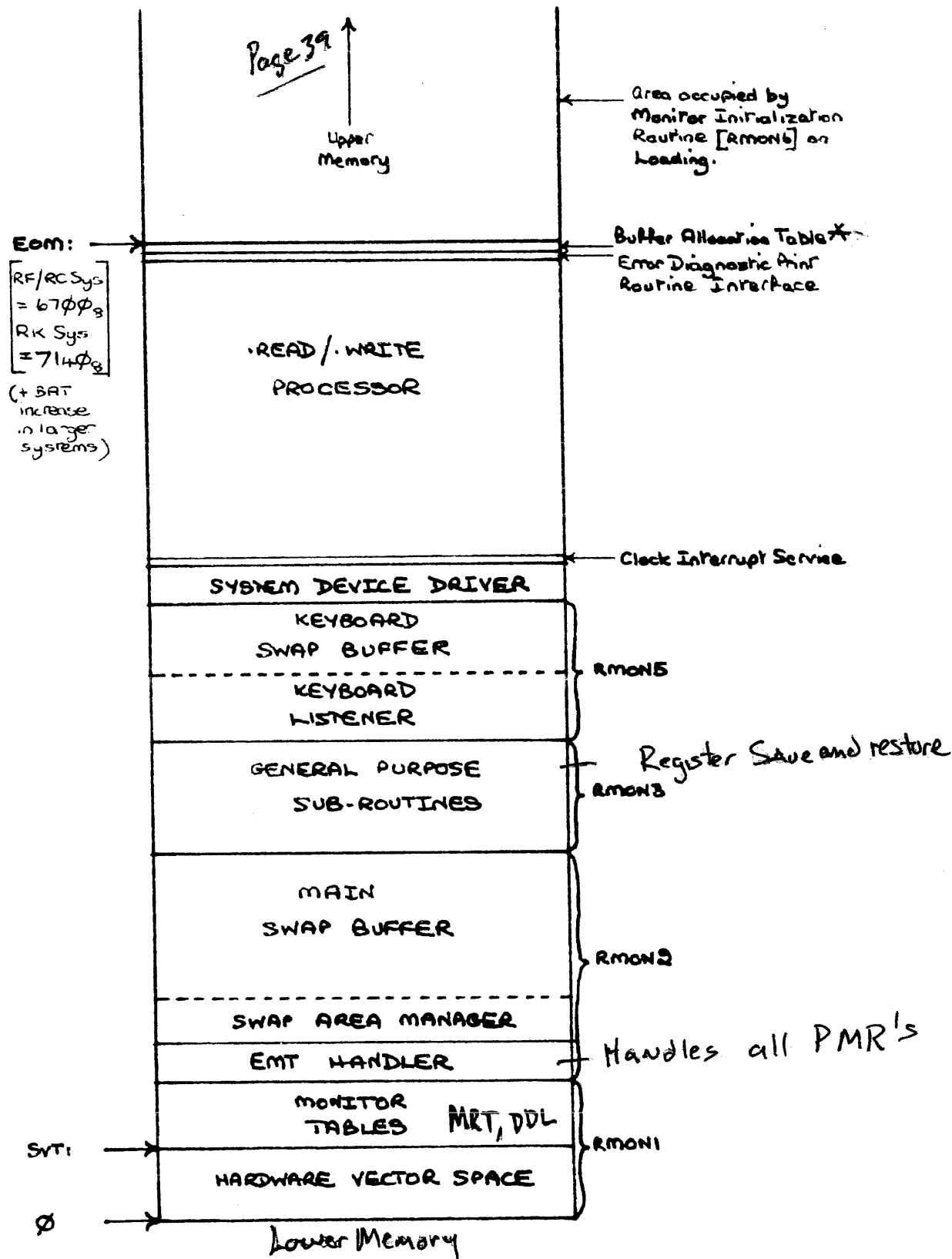
DEVICE NAME (Radix-50)	
MEMORY START ADDRESS OR 0 = zero means	
INTERRUPT VECTOR ADDRESS	
SIZE IN 16-word UNITS	START BLOCK ON SYSTEM-DEVICE

on the disk

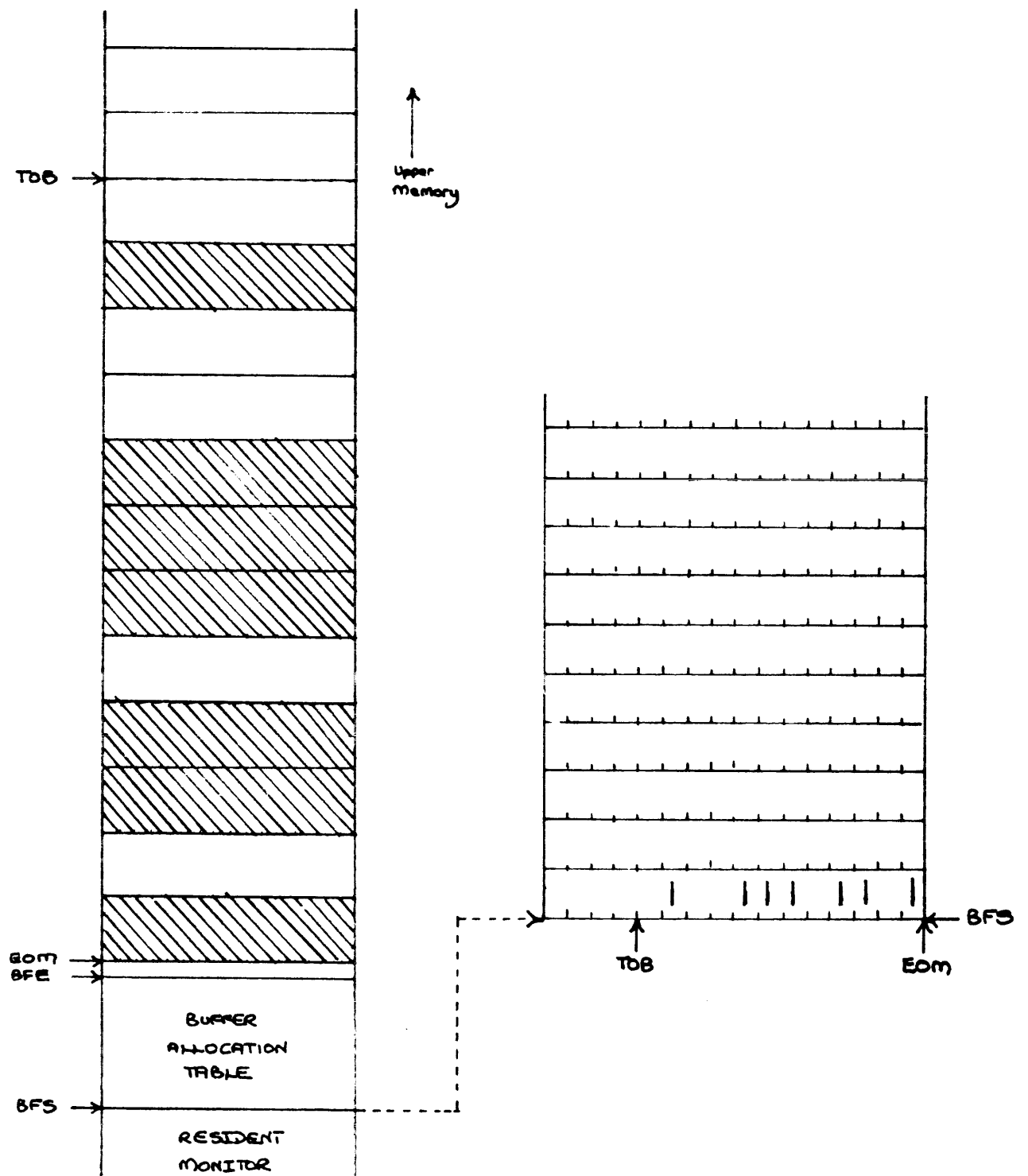
.. DEVICE DRIVER LIST FORMAT.

The resulting format of the DDL is illustrated at Page 8.
The significance of each item is as follows:

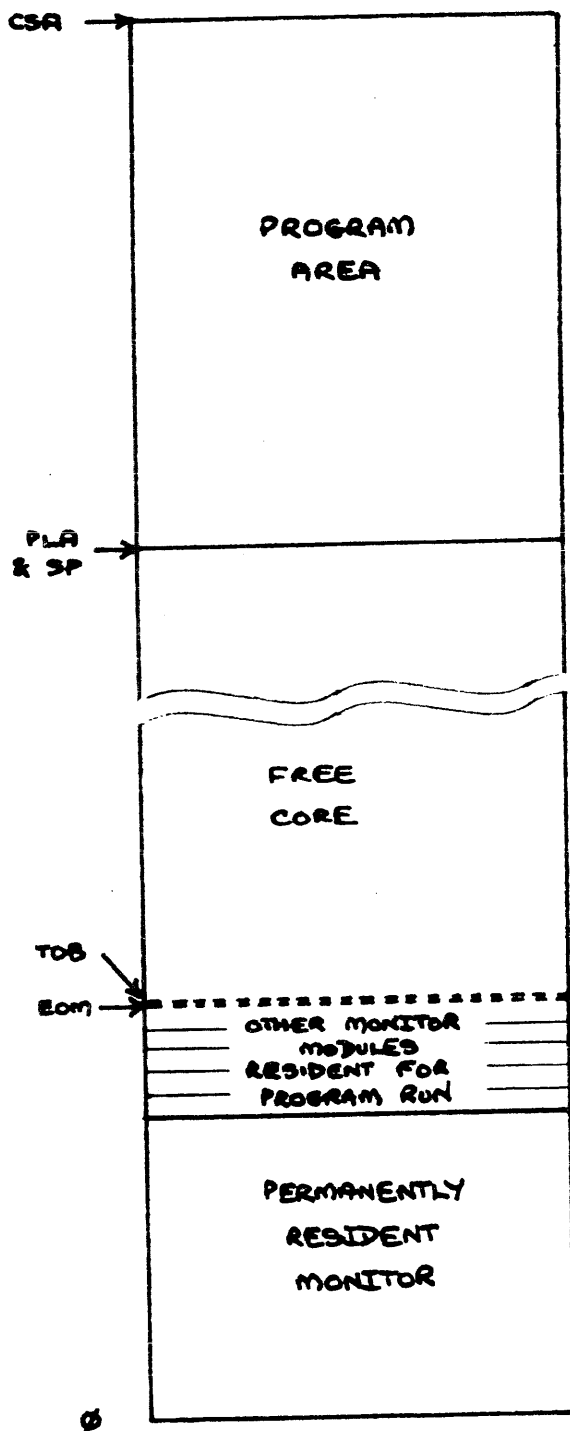
1. Device name - is the radix-5Ø form for the alphanumeric code assigned to each device. The only way an entry in the list can be found is by a search. This item therefore enables its identification.
2. Core load address - contains the start of the device driver when in core. Otherwise Ø indicates the driver's current non-residency.
3. Interrupt vector address - is the start address of the two-word vector assigned to the device within memory locations Ø-377 (or as otherwise provided) and is needed to permit the linkage of the driver's interrupt service routine to the vector when loaded. (See Section 3.2.1) it is held within the DDL rather than the driver itself, because the user can physically reassign devices to different vectors. This obviates reassembly of the driver and also allows in-core modification, even though the driver at the time is available only in the external Monitor Library.
4. System-device start block - gives the actual device address for the driver within the Monitor Library. As for the MRT, no driver is expected to start beyond block #1777(8).
5. # of 16-word blocks - enables determination of the size of the driver for claiming and releasing the buffer it occupies while in core and for specifying of # of words to be read, in order to perform the necessary load (see Section 3.2.1). The six-bit capacity of the field allows a maximum driver length of 1K, which is more than ample in most cases.
6. DDL end - replaces the external address data in the system-device entry in the table since the latter information is irrelevant when, by its purpose, the driver for such device can never leave memory. The provision of this item instead allows the DDL to be of variable size and hence contain only the drivers needed to support the particular configuration being used. However it is essential that the system device entry is always the first in the DDL as shown in the diagram.



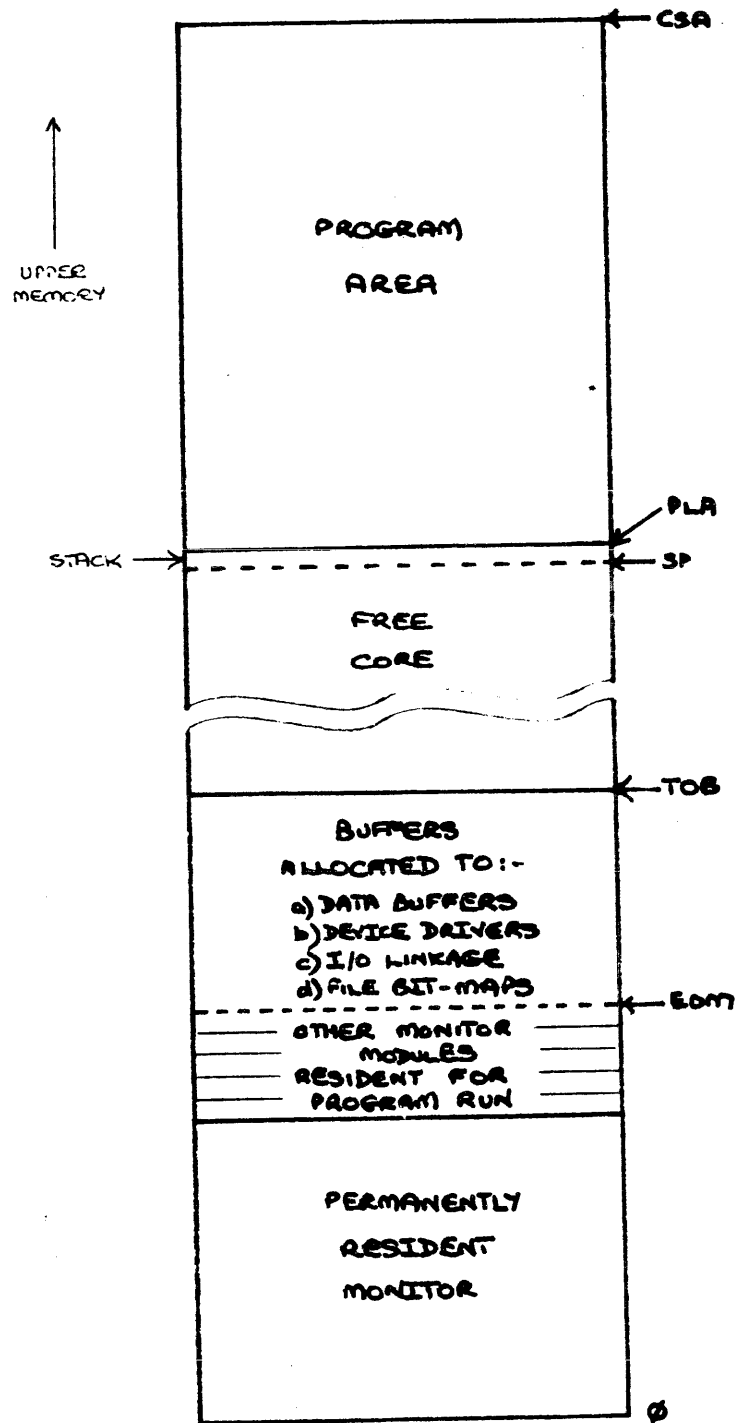
* Each bit controls 16 words of free core area
THE BASIC RESIDENT MONITOR in memory



Buffer Allocation Management



memory after
Program load



Memory during
Program Run.

63 VIC'S

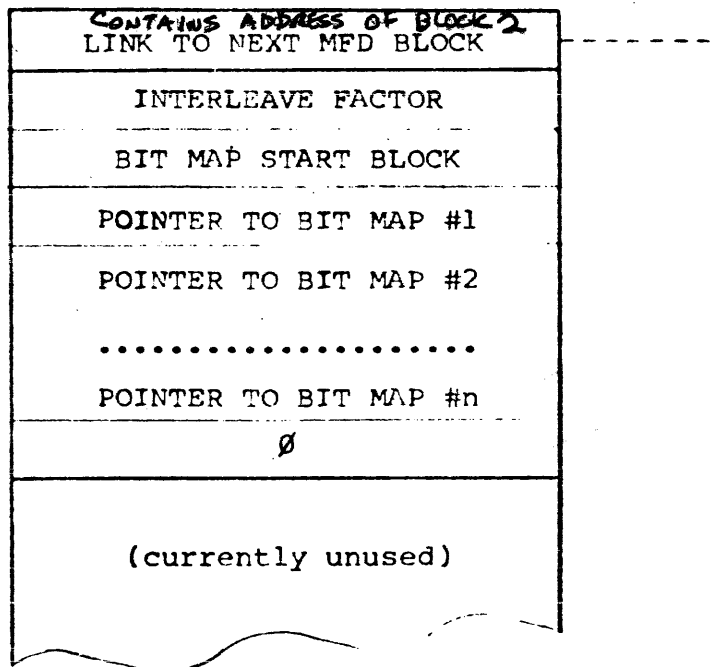


Fig.4-3: Master File Directory Block #1

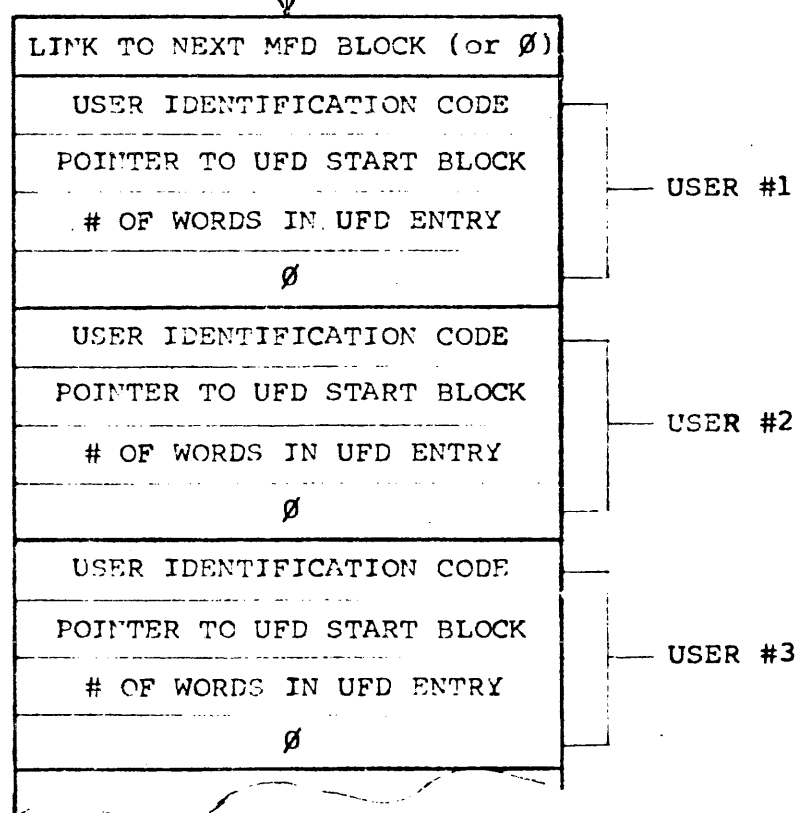


Fig.4-4: Master File Directory Block #2

LINK TO NEXT UFD BLOCK (or 0)		
FILE		
NAME		
EXTENSION		
TYPE	* 0	(resvd) CREATION DATE
NEXT FREE BYTE		
START BLOCK #		
LENGTH # OF BLOCKS		
LAST BLOCK WRITTEN IN BLOCKS		
LOCK	USAGE COUNT	PROTECTION CODE
FILE		
NAME		
EXTENSION		
TYP.	>	(resvd) CREATION DATE
NEXT FREE BYTE		
START BLOCK #		
LENGTH		
LAST BLOCK WRITTEN		
LOCK	USAGE COUNT	PROTECTION CODE
FILE		
NAME		
EXTENSION, etc		

FILED IN AFTER DATA TRANSFER HAS BEEN CLOSED

9 WORDS

FILE ENTRY #1

FILE ENTRY #2

User File Directory Block

* 1 = LINKED
0 = CONTIGUOUS

LINK TO NEXT MAP BLOCK	
MAP #	
# OF WORDS OF MAP	
LINK TO FIRST MAP BLOCK	
MAP FOR BLOCKS 0 - 17	
- " -	20 - 37
- " -	40 - 57
- " -	60 - 77
- " -	100 - 117
- " -	120 - 137
- " -	140 - 157
- " -	160 - 177
- " -	200 - 217
- " -	210 - 237
etc.	

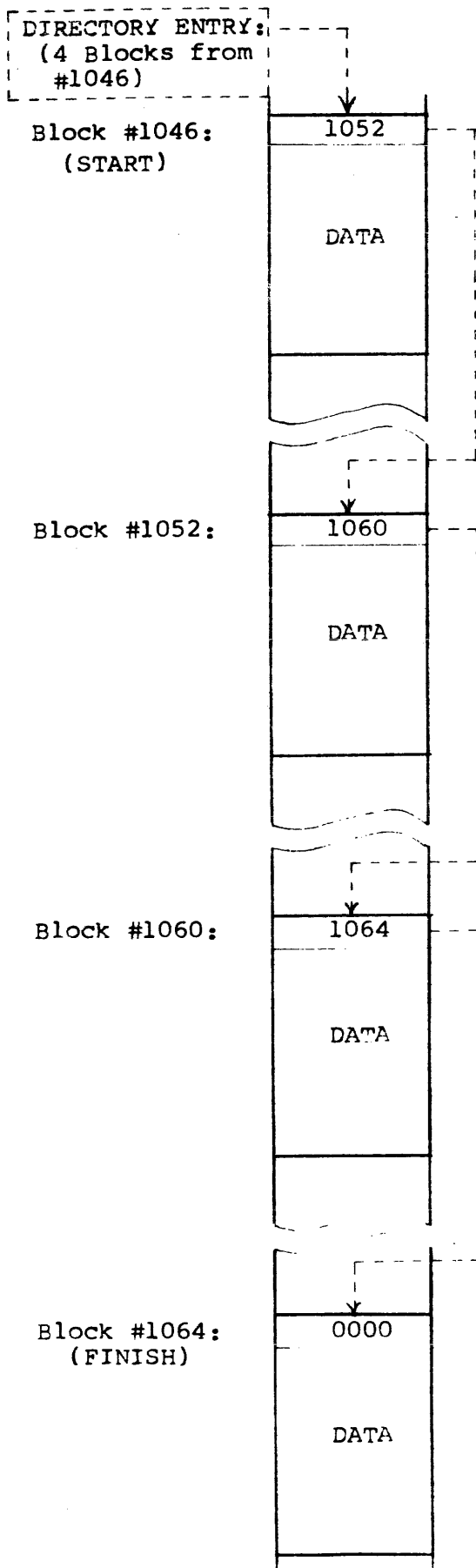
960,108 blocks

77	76	75	74	73	72	71	70	67	66	65	64	63	62	61	60
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

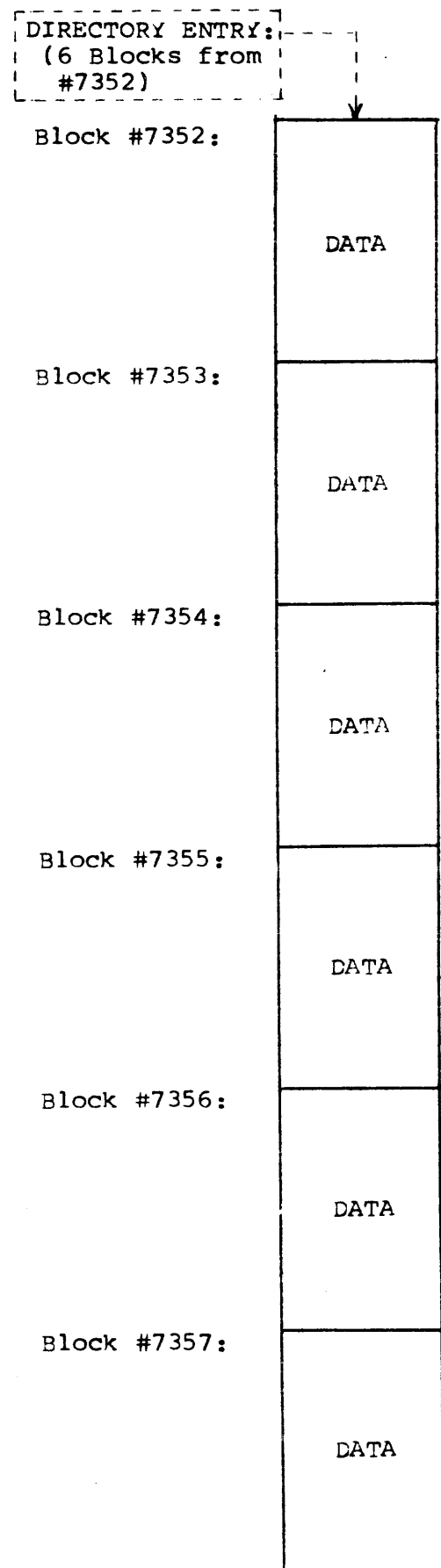
Bit-map Segment Format

OWNER	GROUP	OTHERS
-------	-------	--------

Protection Code Format



Linked File Format.



Contiguous File Format.

(Reserved for Bootstrap)
MFD BLOCK #1
USER #1 - UFD #1
USER LINKED FILES & OTHER UFD BLOCKS USER CONTIGUOUS FILES
MFD BLOCK #2
BIT MAP BLOCK #1
.....
BIT MAP BLOCK #n

BLOCK		
...	0	...
...	1	...
...	2	...
...	3	...
	4	...

Hardware
Protect
Line →

SYSTEM BOOTSTRAP
MFD BLOCK #1
SYSTEM UFD BLOCK #1
SYSTEM UFD BLOCK #2
MONITOR LIBRARY SYSTEM PROGRAMS USER LINKED FILES & OTHER UFD BLOCKS USER CONTIGUOUS FILES LINK.DB SYSTEM PROGRAM OVERLAYS
MFD BLOCK #2
BIT MAP BLOCK #1
.....
BIT MAP BLOCK #n

MASTER FILE DIRECTORY

Non-system Disk Format

System Disk Format

Only 56 unique file names on a DECTape

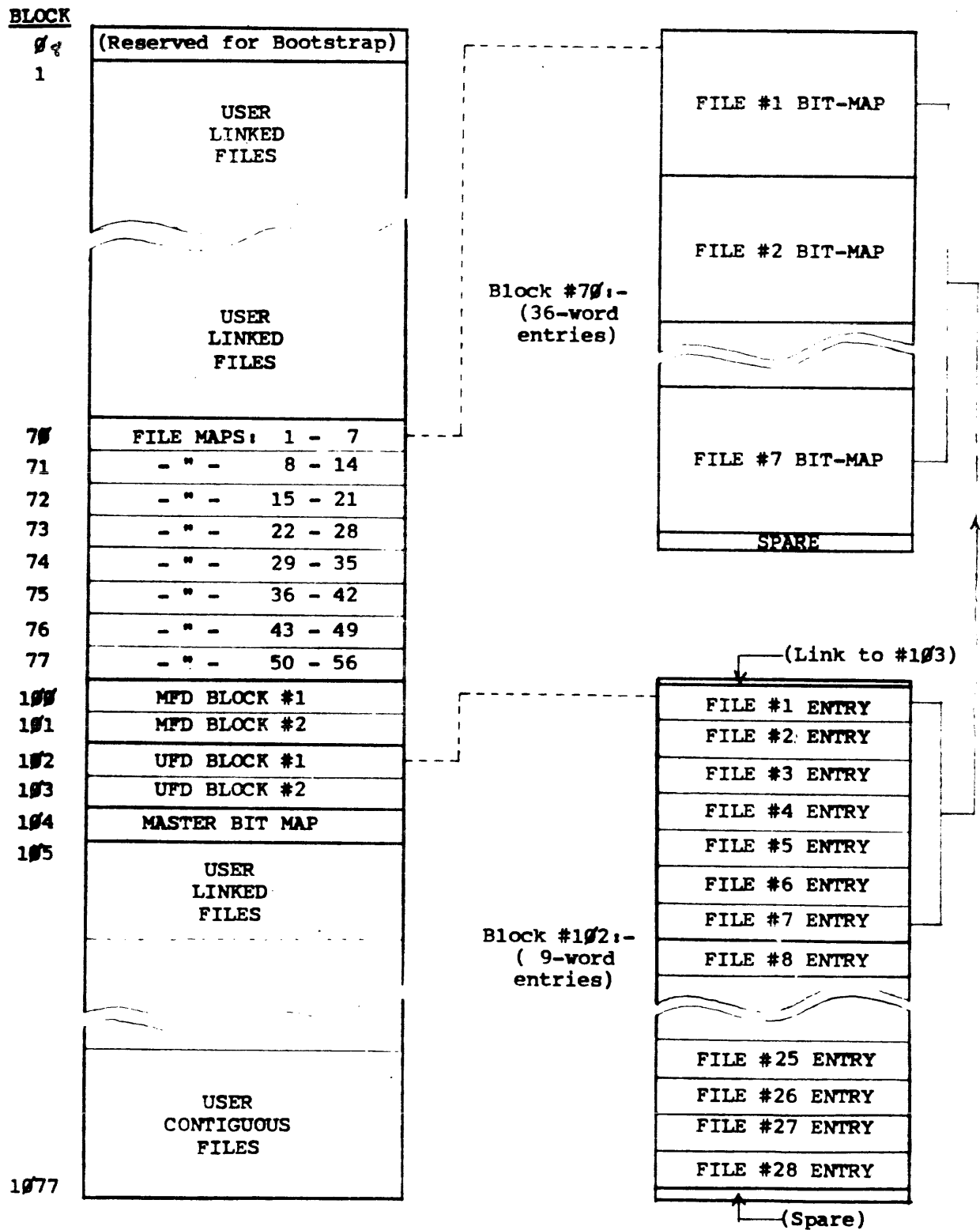


Fig.4-11: DECTape Format

576 Blocks on A DECTAPE

DOS MACRO-11

I. General Assembler Directives

A. Listing Control Directives

1. .LIST *arg*
2. .NLIST *arg*

Arguments:	Controls listing of:
SEQ	source line sequence numbers
LOC	location counter
BIN	Binary output
BEX	binary extensions of instructions
SRC	Source code
COM	comments
MD	macro definitions and repeat
MC	range expansions
* ME	macro calls and repeat range
* MEB	expansions
CND	macro expansions
* LD	macro expansion binary code
TOC	unsatisfied conditions, .IF
TTM	and .ENDC statements
SYM	listing directives without
	arguments
	table of contents on assembly
	pass 1
	listing output format
	symbol table for the assembly
*Default = NOLIST	

3. Switches on the listing file specification during assembly:

/LI:arg
/NL arg

B. Page Headings

1. TITLE
2. .SBTTL
3. .IDENT
4. .PAGE

C. Functions

1. .ENABL arg
2. .DSABL arg

Arguments	Description
ABS	outputs in absolute binary
AMA	converts all mode 67's to 37
CDR	columns 73+ are treated as comments
FPT	causes floating point truncation
LC	accepts lower case and converts to upper case
LSB	enables or disables a local symbol block
PNC	enables or disables binary output

3. Switches:

- /EN: arg
- /DS: arg

D. Data Storage

1. .BYTE
2. .WORD
3. '
4. "
5. .ASCII
6. .ASCIIZ
7. .RAD50

E. Radix Control

1. .RADIX
2. ↑D
3. ↑O
4. ↑B

F. Location Counter Control

1. .EVEN
2. .ODD
3. .BLKB
4. .BLKW

G. Numeric Control

1. .FLT2
2. .FLT4
3. ↑F
4. ↑C

H. Terminating Directives

1. .END
2. .EOT

I. Program Boundaries

1. .LIMIT

J. Program Sectioning

1. .ASECT
2. .CSECT
3. .CSECT Symbol

K. Symbol Control

1. .GLOBL

L. Conditional Assemblies

1. IF ...

II. MACRO Directives

A. MACRO Definition

1. .MACRO name, dummy arg...
2. .ENDM
3. .MEXIT
4. MACRO calls
5. Concatenation
6. .NARG
7. .NCHR
8. .TYPE
9. .ERROR
10. .PRINT
11. .IRP
12. .IRPC
13. .REPT
14. .MCALL

```

1          ,TITLE DEMO
2
3          000000 R0=    %0
4          000001 R1=    %1
5          000002 R2=    %2
6          000003 R3=    %3
7          000004 R4=    %4
8          000005 R5=    %5
9          000006 SP=    %6
10         000007 PC=    %7
11
12         ,LIST MEB          ;LIST MACRO EXPANSIONS OF LINES
13                             ;WHICH GENERATE BINARY CODE
14
15         ;SIMPLIFY INSTRUCTION
16
17         ,MACRO CALL ADDR
18         ,GLOBL ADDR
19         JSR PC,ADDR
20         ,ENDM
21
22 000000 CALL SUBR
23 000000 004767 JSR PC,SUBR
24 000000 000000G
25
26         ;HANDLE ARGUMENT LISTS
27
28         ,MACRO FUNCT ARG1, ARG2, ARG3
29         ,GLOBL SUBR
30         JSR PC,SUBR
31         ,WORD ARG1, ARG2, ARG3
32         ,ENDM
33
34 000004 FUNCT 1,2,3
35 000004 004767 JSR PC,SUBR
36 000000 000000G
37 000100 000001 ,WORD 1, 2, 3
38 000102 000002
39 000104 000003
40
41         ;THIS COULD BE REDEFINED AT A LATER DATE TO:
42
43         ,MACRO FUNCT ARG1, ARG2, ARG3
44         MOV #ARG3,-(SP)
45         MOV #ARG2,-(SP)
46         MOV #ARG1,-(SP)
47         TRAP 10
48         ,ENDM
49
50 000106 FUNCT 4,5,6
51 000106 012746 MOV #6,-(SP)
52 000000 000006
53 000202 012746 MOV #5,-(SP)
54 000000 000005
55 000206 012746 MOV #4,-(SP)
56 000000 000004
57 000302 104410 TRAP 10

```

```

1      ;ESTABLISH DEFAULT VALUES WHICH CAN BE
2      ;OVER-RIDDEN IN A PAREMETER FILE
3
4      .MACRO  PARAM  MNE,VALUE
5      .IF NDF MNE,  MNE=  VALUE
6      .LIST      ;FORCE LISTING OF FINAL VALUE
7      MNE=
8      .NLIST
9      .ENDM
10
11 00034      PARAM  ALPHA, 1000.
12      001750 ALPHA= ALPHA
13
14 00034      000200 BETA= 200
15      PARAM  BETA,120
16      000200 BETA= BETA
17
18      ;GENERATE MESSAGE
19
20      .MACRO  GENMSG  STRING,?LABEL
21      .WORD  LABEL-,-2
22      .ASCII 0STRING0
23      .LIST      ;LIST FOR EXAMPLE
24      LABEL: .EVEN
25      .NLIST      ;RETURN TO NORMAL LEVEL
26      .ENDM
27
28 00034      GENMSG  <NOW IS>
29 00034 000006 .WORD  64$-,-2
30 00036      116 .ASCII 0NOW IS0
31 00037      117
32 00040      127
33 00041      040
34 00042      111
35 00043      123
36 00044      64$: .EVEN
37
38      .MACRO  GENMSG  STRING,LABEL
39      .ASCIIZ 0STRING0
40      .EVEN
41      .ENDM
42
43 00044      GENMSG  <THE TIME>
44 00044      124 .ASCIIZ 0THE TIME0
45 00045      110
46 00046      105
47 00047      040
48 00050      124
49 00051      111
50 00052      115
51 00053      105
52 00054      000

```

```

1      .MACRO .PARAM
2      R0=%A00
3      R1=%A01
4      R2=%A02
5      R3=%A03
6      R4=%A04
7      R5=%A05
8      R6=%A06
9      R7=%A07
10     SP=%A06
11     PC=%A07
12     PSW=%A0177776
13     SWR=%A0177570
14     .ENDM
15     .MACRO .INIT .LBLCK
16     .MCALL .AMODE
17     .AMODE .LBLCK
18     EMT <A06>
19     .ENDM
20
21     .MACRO .RLSE .LBLCK
22     .MCALL .AMODE
23     .AMODE .LBLCK
24     EMT <A07>
25     .ENDM
26
27     .MACRO .CLOSE .LBLCK
28     .MCALL .AMODE
29     .AMODE .LBLCK
30     EMT <A017>
31     .ENDM
32
33     .MACRO .READ .LBLCK,.LBUFF
34     .MCALL .AMODE
35     .AMODE .LBUFF
36     .AMODE .LBLCK
37     EMT <A04>
38     .ENDM
39
40     .MACRO .WRITE .LBLCK,.LBUFF
41     .MCALL .AMODE
42     .AMODE .LBUFF
43     .AMODE .LBLCK
44     EMT <A02>
45     .ENDM
46
47     .MACRO .OPENO .LBLCK,.FBLCK
48     .MCALL .CODE,.OPEN
49     .CODE .FBLCK,<A02>
50     .OPEN .LBLCK,.FBLCK
51     .ENDM
52
53     .MACRO .OPENI .LBLCK,.FBLCK
54     .MCALL .CODE,.OPEN
55     .CODE .FBLCK,<A04>
56     .OPEN .LBLCK,.FBLCK
57     .ENDM

```

```
58
59      .MACRO      .OPENU  .LBLCK,.FBLCK
60      .MCALL      .CODE,.OPEN
61      .CODE        .FBLCK,<A01>
62      .OPEN        .LBLCK,.FBLCK
63      .ENDM
64
65      .MACRO      .OPENC  .LBLCK,.FBLCK
66      .MCALL      .CODE,.OPEN
67      .CODE        .FBLCK,<A013>
68      .OPEN        .LBLCK,.FBLCK
69      .ENDM
70
71      .MACRO      .OPENE  .LBLCK,.FBLCK
72      .MCALL      .CODE,.OPEN
73      .CODE        .FBLCK,<A03>
74      .OPEN        .LBLCK,.FBLCK
75      .ENDM
76
77      .MACRO      .OPEN    .LBLCK,.FBLCK
78      .MCALL      .AMODE
79      .AMODE      .FBLCK
80      .AMODE      .LBLCK
81      EMT <A016>
82      .ENDM
83
84
```

```

1
2      .MACRO   .WAIT   .LBLCK
3      .MCALL   .AMODE
4      .AMODE   .LBLCK
5      EMT <A01>
6      .ENDM
7
8      .MACRO   .WAITR   .LBLCK, .ADDR
9      .MCALL   .AMODE
10     .AMODE   .ADDR
11     .AMODE   .LBLCK
12     EMT <A00>
13     .ENDM
14
15     .MACRO   .BLOCK   .LBLCK, .BBLCK
16     .MCALL   .AMODE
17     .AMODE   .BBLCK
18     .AMODE   .LBLCK
19     EMT <A011>
20     .ENDM
21
22     .MACRO   .TRAN    .LBLCK, .TBLCK
23     .MCALL   .AMODE
24     .AMODE   .TBLCK
25     .AMODE   .LBLCK
26     EMT <A010>
27     .ENDM
28
29     .MACRO   .SPEC    .LBLCK, .SARG
30     .MCALL   .AMODE
31     .AMODE   .SARG
32     .AMODE   .LBLCK
33     EMT <A012>
34     .ENDM
35
36     .MACRO   .STAT    .LBLCK
37     .MCALL   .AMODE
38     .AMODE   .LBLCK
39     EMT <A013>
40     .ENDM
41
42     .MACRO   .ALLOC   .LBLCK, .FBLCK, .N
43     .MCALL   .AMODE
44     .AMODE   .N
45     .AMODE   .FBLCK
46     .AMODE   .LBLCK
47     EMT <A015>
48     .ENDM
49
50     .MACRO   .DELET   .LBLCK, .FBLCK
51     .MCALL   .AMODE
52     .AMODE   .FBLCK
53     .AMODE   .LBLCK
54     EMT <A021>
55     .ENDM
56
57     .MACRO   .RENAM   .LBLCK, .OFB, .NFB

```

58	.MCALL	.AMODE
59	.AMODE	.NFR
60	.AMODE	.OFB
61	.AMODE	.LBLCK
62	END <A020>	
63	.ENDM	

```

1
2      .MACRO  .APPND  .LBLCK,.1FB,.2FB
3      .MCALL  .AMODE
4      .AMODE  .2FB
5      .AMODE  .1FB
6      .AMODE  .LBLCK
7      EMT <A022>
8      .ENDM
9
10     .MACRO  .LOOK   .LBLCK,.FBLCK,.OP
11     .MCALL  .AMODE
12     .AMODE  .FBLCK
13     .IIF NB,.OP,CLR -(SP)
14     .AMODE  .LBLCK
15     EMT <A014>
16     .ENDM
17
18     .MACRO  .KEEP    .LBLCK,.FBLCK
19     .MCALL  .AMODE
20     .AMODE  .FBLCK
21     .AMODE  .LBLCK
22     EMT <A024>
23     .ENDM
24
25     .MACRO  .EXIT
26     EMT <A060>
27     .ENDM
28
29     .MACRO  .TRAP    .STUS,.ADDR
30     .MCALL  .AMODE
31     .AMODE  .ADDR
32     .AMODE  .STUS
33     MOV     #A01,-(SP)
34     EMT <A041>
35     .ENDM
36
37     .MACRO  .STFPU   .STUS,.ADDR
38     .MCALL  .AMODE
39     .AMODE  .ADDR
40     .AMODE  .STUS
41     MOV     #A03,-(SP)
42     EMT <A041>
43     .ENDM
44
45     .MACRO  .RECRD   .LBLCK,.RBLCK
46     .MCALL  .AMODE
47     .AMODE  .RBLCK
48     .AMODE  .LBLCK
49     EMT <A025>
50     .ENDM
51
52     .MACRO  .DUMP     .LOW,.HIGH,.CDE
53     .MCALL  .AMODE
54     .AMODE  .LOW
55     .AMODE  .HIGH
56     .AMODE  .CDE
57     EMT <A064>

```

```
58 .ENDM
59
60 .MACRO .RSTRT .ADDR
61 .MCALL .AMODE
62 .AMODE .ADDR
63 MOV #A02,-(SP)
64 EMT <A041>
65 .ENDM
66
67 .MACRO .CORE
68 MOV #A0100,-(SP)
69 EMT <A041>
70 .ENDM
71
72 .MACRO .MONR
73 MOV #A0101,-(SP)
74 EMT <A041>
75 .ENDM
76
77 .MACRO .MONF
78 MOV #A0102,-(SP)
79 EMT <A041>
80 .ENDM
81
82 .MACRO .DATE
83 MOV #A0103,-(SP)
84 EMT <A041>
85 .ENDM
```

```

1
2      .MACRO .TIME
3      MOV    #A0104,-(SP)
4      EMT <A041>
5      .ENDM
6
7      .MACRO .GTUIC
8      MOV    #A0105,-(SP)
9      EMT <A041>
10     .ENDM
11
12     .MACRO .SYSDV
13     MOV    #A0106,-(SP)
14     EMT <A041>
15     .ENDM
16
17     .MACRO .RADPK .ADDR
18     .MCALL .AMODE
19     .AMODE .ADDR
20     CLR    -(SP)
21     EMT <A042>
22     .ENDM
23
24     .MACRO .RADUP .ADDR,.WRD
25     .MCALL .AMODE
26     .AMODE .WRD
27     .AMODE .ADDR
28     MOV    #A01,-(SP)
29     EMT <A042>
30     .ENDM
31
32     .MACRO .D2BIN .ADDR
33     .MCALL .AMODE
34     .AMODE .ADDR
35     MOV    #A02,-(SP)
36     EMT <A042>
37     .ENDM
38
39     .MACRO .BIN2D .ADDR,.WRD
40     .MCALL .AMODE
41     .AMODE .WRD
42     .AMODE .ADDR
43     MOV    #A03,-(SP)
44     EMT <A042>
45     .ENDM
46
47     .MACRO .O2BIN .ADDR
48     .MCALL .AMODE
49     .AMODE .ADDR
50     MOV    #A04,-(SP)
51     EMT <A042>
52     .ENDM
53
54     .MACRO .BIN2O .ADDR,.WRD
55     .MCALL .AMODE
56     .AMODE .WRD
57     .AMODE .ADDR

```

.MAIN. MACRO VR05A 31-OCT-72 17:56 PAGE 5+

58
59
60

MOV #A05, -(SP)
EHT <A042>
.ENDM

```

1
2      .MACRO .CSI1 .CMDBF
3      .MCALL .AMODE
4      .AMODE .CMDBF
5      EMT <A056>
6      .ENDM
7
8      .MACRO .CSI2 .CSBLK
9      .MCALL .AMODE
10     .AMODE .CSBLK
11     EMT <A057>
12     .ENDM
13
14     .MACRO .DTCVT .ADDR
15     .MCALL .CVTDT
16     .CVTDT #A00,.ADDR
17     .ENDM
18
19     .MACRO .TMCVT .ADDR
20     .MCALL .CVTDT
21     .CVTDT #A01,.ADDR
22     .ENDM
23
24     .MACRO .CVTDT .CDE,.ADDR,.VAL1,.VAL2
25     .MCALL .AMODE
26     .IF NB,.VAL2
27     .AMODE .VAL2
28     .ENDC
29     .IF NB,.VAL1
30     .AMODE .VAL1
31     .ENDC
32     .AMODE .ADDR
33     .AMODE .CDE
34     EMT <A066>
35     .ENDM
36
37     .MACRO .GTPLA
38     CLR -(SP)
39     MOV #A05,-(SP)
40     EMT <A041>
41     .ENDM
42
43     .MACRO .STPLA .ADDR
44     .MCALL .AMODE
45     .AMODE .ADDR
46     MOV #A05,-(SP)
47     EMT <A041>
48     .ENDM
49
50     .MACRO .GTCIL
51     MOV #A0111,-(SP)
52     EMT <A041>
53     .ENDM
54
55     .MACRO .GTSTK
56     CLR -(SP)
57     MOV #A04,-(SP)

```

```

58          EMT <A041>
59          .ENDM
60
61          .MACRO .STSTK .ADDR
62          .MCALL .AMODE
63          .AMODE .ADDR
64          MOV     #A04,-(SP)
65          EMT <A041>
66          .ENDM
67
68          .MACRO .RUN .RNBLK
69          .MCALL .AMODE
70          .AMODE .RNBLK
71          EMT <A065>
72          .ENDM
73
74          .MACRO .FLUSH .CDE
75          .MCALL .AMODE
76          .AMODE .CDE
77          EMT <A067>
78          .ENDM
79
80          ; THE MACRO .AMODE ACCEPTS ONE ARGUMENT AND
81          ; AS A FUNCTION OF THE ADDRESSING MODE OF
82          ; THE ARGUMENT GENERATES THE APPROPRIATE
83          ; MOV TO -(SP).
84          ; ADDRESS MODES THAT ARE TROUBLESOME (E.G.
85          ; X(SP)) OR UNLIKELY (E.G. SP) WILL RESULT
86          ; IN A .ERROR TO CMO INCLUDING THE
87          ; VALUE OF THE ADDRESS MODE (E.G. X(SP)
88          ; IS REPRESENTED AS 000066), THE ARGUMENT ITSELF
89          ; AND THE TEXT "ADDRESSING MODE ILLEGAL AS SYSTEM
90          ; MACRO ARGUMENT".
91          ;
92          .MACRO .AMODE .ARG
93          SP=%A06
94          .NTYPE .SYM,.ARG          ;.SYM=ADDRESS MODE.
95
96          .IF LE,.SYM-A05
97          MOV     .ARG,-(SP)          ;R0 TO R5
98          .MEXIT
99          .ENDC
100
101          .IF EQ,.SYM&A070-A010
102          .IF LE,.SYM&A07-A06
103          MOV     .ARG,-(SP)          ;R0 TO R6
104          .MEXIT
105          .ENDC
106          .ENDC
107
108          .IF EQ,.SYM&A060-A020
109          MOV     .ARG,-(SP)          ;[0](R0)+ TO [0](R7)+
110          .MEXIT                      ; #N,#ADDR
111          .ENDC
112
113          .IF EQ,.SYM&A040-A040
114          .IF LE,.SYM&A07-A05

```

```

115      MOV      .ARG, -(SP)      ;[0]-(R0) TO [0]-(R5)
116      .MEXIT      ;[0]X(R0) TO [0]X(R5)
117      .ENDC
118      .ENDC
119
120      .IF EQ, .SYM&A067-A067
121      MOV      .ARG, -(SP)      ;ADDR AND *ADDR
122      .MEXIT
123      .ENDC
124
125      .ERROR .SYM      ;.ARG ADDRESSING MODE ILLEGAL
126      .PRINT      ;AS SYSTEM MACRO ARGUMENT.
127      .ENDM
128
129      ; THE MACRO .CODE SETS UP THE FILEBLOCK
130      ; WITH THE HOW OPEN CODE.
131      ; THE ADDRESS OF THE FILEBLOCK MUST
132      ; BE IN A REGISTER (R0 TO R5)
133
134      .MACRO .CODE .FBLK, .N
135      .NTYPE .SYM, .FBLK
136
137      .IF LE, .SYM-A05
138      MOVB      #.N, -A02(.FBLK) ;R0 TO R5
139      .MEXIT
140      .ENDC
141
142      .ERROR .SYM      ;.FBLK ADDRESSING MODE ILLEGAL
143      .PRINT      ;FOR .OPEN FILE BLOCK
144      .ENDM
145
146 ***** E

```

LNKBLK:

ERROR RETURN ADDRESS (no buffers)	
Ø or Set to <u>DD</u> B ADDRESS by .INIT	
DATASET LOGICAL NAME *	
DEVICE UNIT #	# of words to follow
DEVICE NAME*	

*=Packed in
Radix-5Ø format

EXTENSIBLE TO PROVIDE SWITCH SPACE IF
THE COMMAND STRING INTERPRETER IS BEING
USED.

(Size indicated in "# of words to
follow")

EXAMPLE

.WORD Ø

LNKBLK :Ø

.RAD50 /ABC/

.BYTE 1, Ø

.RAD50 /DT/

LINK BLOCK

MONITOR LINK = Points to next DDB in chain		IF LAST will equal 0
DRIVER QUEUE LINK -		
DRIVER ROUTINE INDEX	PRIORITY LEVEL IN Q	
Associated DRIVER ADDRESS		SET UP INIT
BUSY FLAG (0 = Idle) CAUSE TO WAIT		
USER LINE ADDRESS		TEMP ADDR OR LINE BUFFER ADDR
DEVICE BLOCK #		
MEMORY BUFFER ADDRESS		INTERNAL
BUFFER WORD COUNT		2's COMPLEMENT # of WORDS to transfer
STATUS		FORMAT
COMPLETION RETURN		CLEAN UP ROUTINE
DRIVER WORD COUNT		# OF WORDS NOT TRANSFERRED in 2's complement
BYTE COUNT		
CHECKSUM STORE		FORMATTED BINARY
POMTER TO DAT ADDRESS of ASSIGNMENT ENTRY		
F.I.B. LINK		FILE STRUCTURED OPERATIONS

P.35 DDB: FROM LINK BLOCK

DATASET DATA-BLOCK
(DDB)

LOGICAL NAME OF DATASET [#] concerned	
DEVICE - NAME *	
DEVICE UNIT #	# OF WORDS FOLLOWING
FILE -	
NAME *	
FILE EXTENSION*	
USER IDENTIFICATION CODE	

= PACKED in
Radix-59 Format

DEVICE ASSIGNMENT
TABLE ENTRY FORMAT

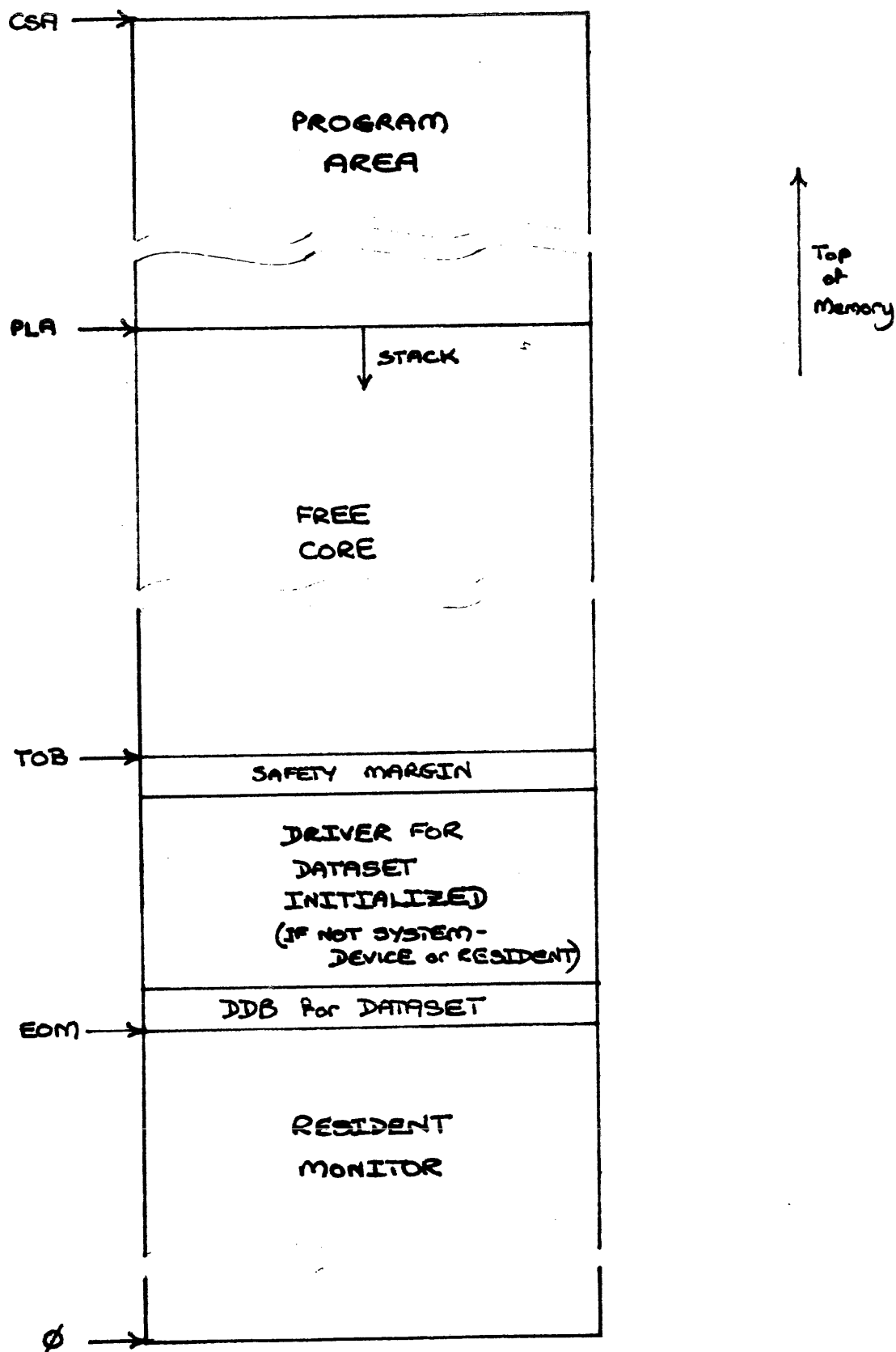
Dataset Data Block

The Monitor has no knowledge of the program's I/O requirements until the program itself identifies these by requesting initialization of the datasets to be used. Hence, as stated in the Programmer's Handbook, .INIT must be the first function called in each case. In response to such call, the Monitor sets up its own control block for the dataset within a 16-word buffer unit claimed from free core and stores its address in the link-word of the user Link-block. This block known as the Dataset Data Block (DDB), is used thereafter as the means by which different Monitor routines pass information to each other and also communicate with the driver servicing the dataset. It is retained in memory until the program releases the dataset from further I/O action, its buffer space is then returned to free core.

The format of the DDB is illustrated on Page 35. The purpose of the items is as follows:

1. Monitor link - enables the Monitor to maintain a control chain of the DDB's currently established. as illustrated. The chain originates at a DDB Chain Origin (DCO) word in the SVT.
2. Q link - is used to chain the DDB's for datasets waiting upon the services of the same device as described in the next section.
3. Priority level - stores the level at which a queued call to the driver is made.
4. Driver routine index - is normally $\emptyset$ unless a driver call is queued: a pointer to the driver routine required is then saved here instead.
5. Driver address - contains the address from which the driver associated with the dataset is loaded. This also serves to identify the driver when compared with entries in the DDL.
6. Busy flag - is set by the EMT Handler to the address of the program I/O call when accepted. This forces any subsequent call to wait until the current request has been satisfied - when it is reset to $\emptyset$. The address of this word is the one stored in the user Link-block as the DDB connection to its dataset.

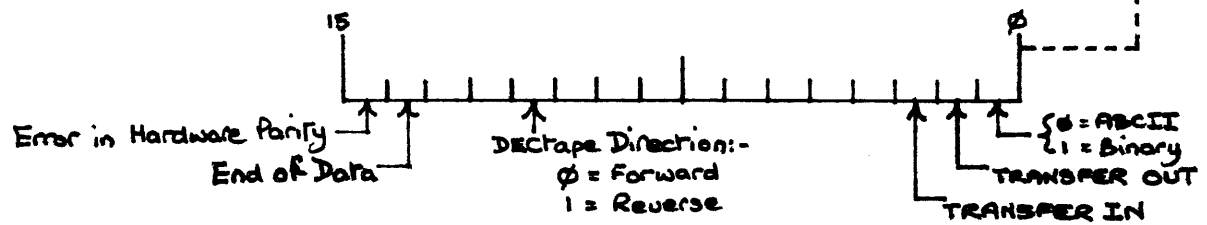
7. User line address - in general is used to save the second parameter passed by most I/O calls (normally the address of a data-block in the program, e.g.. Line Buffer in .READ/.WRITE, TRA-block in .TRAN, etc.
8. Device block - points to the device address of a block on a bulk storage medium.
9. Buffer address - shows the start of a memory area for transfer - normally this is an internal Monitor buffer; thus this word is also used as an indication of the current assignment of such buffer (cleared to $\emptyset$ if none).
10. Word count - gives the number of words to be transferred by the device as a two's complement value.
11. Status - is used to direct the driver on the type of transfer required, allows the driver to return error indicators and stores other control information on a bit-basis.
12. Completion return - shows the address at which a calling routine requires the driver to return when a requested service has been satisfied.
13. Driver word count - allows the driver to indicate (again as two's complement) how many words are not transferred because an end of data point is reached. This word is also used to store a variable pointer to the next byte to be processed in the internal Monitor buffer. (No device action can be underway at this time).
14. Byte count - is used during .READ/.WRITE processing to control bytes passed between the program line and the internal Monitor buffer.
15. Checksum - is mainly provided for the processing of formatted binary data.
16. DAT pointer - is set during dataset initialization to the address of an entry in the Device Assignment Table corresponding to the dataset, if such exists (see previous section).
17. FIB link - connects the DDB to a 16-word buffer extension whenever file-structured operations are underway on the dataset. It is set to $\emptyset$ otherwise.



Memory after Dataset
Initialization

TABLE:

DEVICE BLOCK #
MEMORY START ADDRESS
WORD COUNT (positive)
FUNCTION / STATUS
OF WORDS NOT TRANSFERRED



TRAN-block Format.

OPEN COMMANDS

1. UPDATE

- a. HOW OPEN CODE 1.
- b. Contiguous file only
- c. One block is input from contiguous file into Monitor buffer or output from Monitor buffer to contiguous file.
- d. IO allowed:
 - (1) .BLOCK IN
 - (2) .BLOCK OUT

2. OUTPUT

- a. HOW OPEN CODE 2.
- b. Linked File only
- c. Creates starting block of a linked file.
- d. IO allowed: .WRITE

3. EXTENSION

- a. HOW OPEN CODE 3.
- b. Linked file only
- c. Last Block is Input and New Data is Added Starting in the Next Free Byte. This and ensuing blocks are written into via .WRITE At .CLOSE the Directory is updated to Reflect File Size and New Last Block.
- d. IO allowed: .WRITE

4. INPUT

- a. HOW OPEN CODE 4.
- b. Linked or contiguous file.
- c. First block of specified file is read into Monitor buffer anticipating .READ.
- d. IO allowed:
 - (1) .READ
 - (2) .BLOCK IN

5. CONTIGUOUS

- a. HOW OPEN CODE 13.
- b. Contiguous file only.
- c. Opens a previously created contiguous file for sequential output (.WRITE) so that you may enter data sequentially into physically contiguous blocks.
- d. IO allowed: .WRITE

FILBLK:

ERROR RETURN ADDRESS	
ERROR STATUS	HOW OPEN CODE
FILE-NAME (in Radix-50)	
EXTENSION (in Radix-50)	
USER IDENTIFICATION CODE	
(spare)	PROTECTION CODE

User File Block

DDB + 24
26

(DDB)
DAT ENTRY ADDRESS
POINTER TO FIB

FIB + 0

2
4
6
10
12
14
16
20
22
24
26
30
32
34
36

Information sent to driver	NEXT BLOCK #
	0 HOW OPEN CODE
	EXTENSION START BLOCK #
	TYPE (spare)
	(spare)
	START BLOCK #
	# of BLOCKS
	LAST BLOCK #
	DIRECTORY BLOCK #
	INDEX INTO DIRECTORY BLOCK
	(spare) PROTECTION CODE
	INTERLEAVE FACTOR
	BIT-MAP POINTER
	BIT-MAP Q LINK
	TEMPORARY
	WORK-SPACE

File Information Block

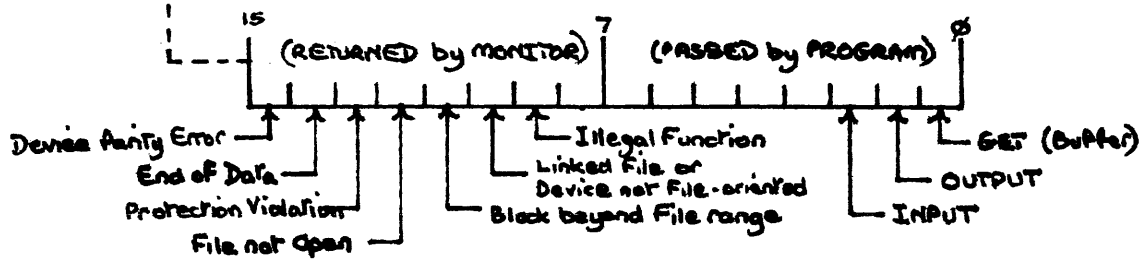
DATA TRANSFER

MAY BE CHANGED
IN V8

BLKBLK:

STATUS	FUNCTION
BLOCK NUMBER (relative)	
MEMORY BUFFER ADDRESS	
LENGTH (words)	

- First Block = $\emptyset$



The BLOCK-block

RECORD Block

The RECORD Block

RECBLK:	FUNCTION / STATUS
	BUFFER ADDRESS
	RECORD LENGTH
	HI ORDER, RECORD #
	LO ORDER, RECORD #

Figure 3-12 The Record Block

ADDRESS

FUNCTION

RECBLK

FUNCTION / STATUS WORD

BIT

- 0 - Not used
- 1 - Record Output - Set by user
- 2 - Record Input - Set by user
- 3-8 - Not used

(Following bits set by Monitor)

- 9 - Illegal Function
- 10 - File is linked or device is not File structured.
- 11 - Record requested lies outside the file.
- 12 - File not OPEN
- 13 - Protect code violation, Incorrect Open
- 14 - Not used
- 15 - Device parity error

The user may set only bits 1 or 2; error bits are set by the Monitor, and should be tested for by the user upon return from the request. The error bits are cleared by the Monitor when a .RECRD request is issued and are set as appropriate upon return from the Monitor.

RECBLK+2

BUFFER ADDRESS

The address of the user's buffer. The buffer must be large enough to contain a record of the length indicated in the next word, as the Monitor assumes that sufficient space is available and will overlay data stored below a buffer of insufficient length.

RECBLK+4

RECORD LENGTH

The number of bytes of a Record. This value, which must remain the same for all records in the file, is supplied by the user.

RECBLK+6

High Order - Record Number

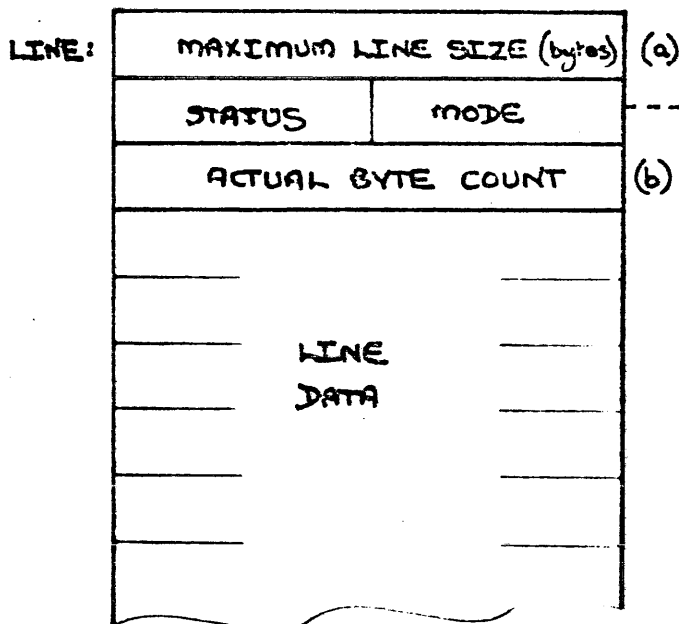
RECBLK+10

Low Order - Record Number

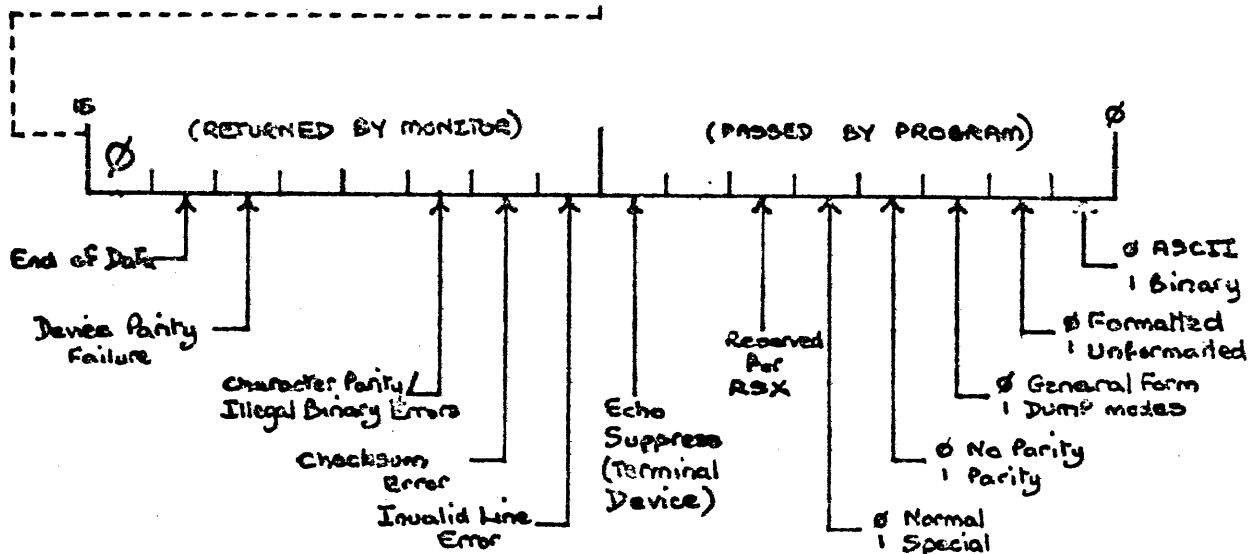
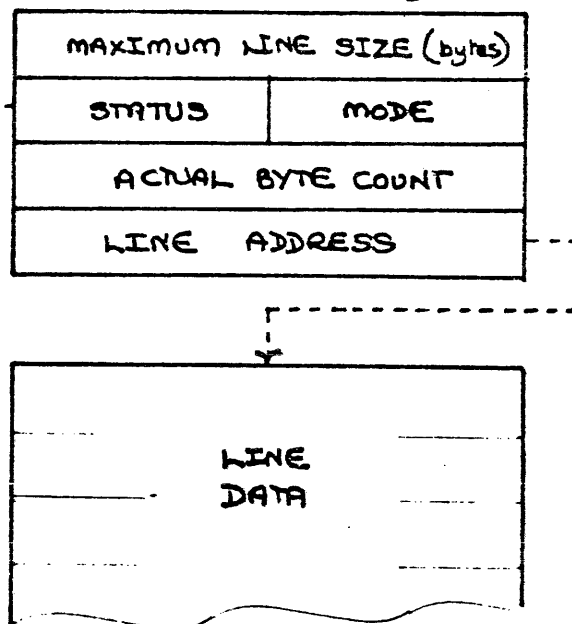
This entry identifies the record to be read or written. Two words are provided in anticipation of files with more than 65,536 records.

First Record of File is number 0.

a) GENERAL FORM:-



b) DUMP MODES only:-



Notes:-

- must be program-supplied for INPUT
- must be program-supplied for OUTPUT

Valid Modes:-

Formatted ASCII Parity (Normal & Special)
 Formatted ASCII Non-parity (Normal & Special)
 Unformatted ASCII Parity / Non-parity (Normal)
 Formatted Binary Non-parity (Normal & Special)
 Unformatted Binary Non-parity (Normal)
 (Dump valid in all above modes)

User LINE Format.

<u>MODE</u>	<u>READ</u> <u>TERMINATION</u>	<u>MAX BYTE</u> <u>COUNT</u>	<u>ACTUAL BYTE</u> <u>COUNT</u>	<u>WRITE</u> <u>TERMINATION</u>	<u>DISCARDED</u> <u>CHARACTERS</u>	<u>BITS</u>
FORMATTED ASCII NORMAL	Line feed, form feed vertical tab, or max Byte count is reached	Invalid line error if max Byte before terminator	Max count plus excess	Actual Byte count-previous terminators out as normal	Rubouts and nulls excess char.	7
FORMATTED ASCII SPECIAL	Line feed, form feed V.T.or max Byte count	Invalid line error if max bytes before terminator	=Max count or less (terminated)	First termina- tor or actual byte count	Rubouts and nulls	7
FORMATTED BINARY NORMAL	Actual Byte count or maxi- mum Byte count	If max. before actual invalid line error	Max count plus excess	Actual Byte count includes words 2+4- checksum calc + output	After max - overlaid in last byte (Read)	8 Bit Transfer
FORMATTED BINARY SPECIAL	Actual Byte count	If max before actual remain- der retained	Max count plus excess	Actual Byte count includes words 2 + 4- checksum calc + output		8 Bit Transfer
UNFORMATTED ASCII NORMAL OR SPECIAL	Maximum byte count			Actual byte count	Nulls only	7 bit transfer Bit 8 set to 0
UNFORMATTED BINARY NORMAL OR SPECIAL	Maximum Byte			Actual byte count no checksum		8 Bit Transfer
FORMATTED ASCII PARITY <i>Uses EVEN Parity</i>	Line feed, form feed vertical tab, or max byte count is reached	Invalid line error if max byte before terminator	See formatted ASCII normal or special	Actual byte count-previous terminators out as normal	See formatted ASCII normal or special	Checks for even parity 7 bit Xfer <u>BAD</u> 8=1
UNFORMATTED ASCII PARITY	Maximum byte count			Actual Byte count	Nulls only	8 Bit transfer

1.Ø ABSTRACT

The TC11 DECTape Formatter is a program for marking DECTapes on Unit Ø with a standard format of 578_{10} blocks of 256_{10} words each.

2.Ø REQUIREMENTS

- A. Minimum configuration PDP-11
- B. TC11 DECTape control and at least one transport.
- C. DECTape(s)

3.Ø LOADING PROCEDURE

The normal procedure for loading Absolute Binary tapes should be followed.

4.Ø PRELIMINARY OPERATIONS

- A. Mount a DECTape on Unit Ø, wrapping enough turns to insure tension - if not enough. Tape will run off reel - Also can have hung read.
- B. Set the transport On-line with Write Enable on.
- C. Set the TC11 WALL and WRTM switches (up). Must Remove Top Cover.

5.Ø STARTING PROCEDURE

Load Address 6000_8 and press START.

6.Ø OPERATING PROCEDURE

- A. Once started the program writes all Timing and Mark Track information in one forward pass and then halts at address 1050_8 .

- B. If the tape has run off the reel, remount it. Reset the WRTM switch (down) and press CONTINUE. The tape will back out for three blocks, then turn around and write the complement obverse of the last block number (11Ø1g) in Writeall in all data slots until the End Zone is sensed. The tape then reverses and writes relevant information in all blocks down through Ø in Writeall.
- C. The tape then reverses and alternately searches for each block number in sequence then changes to Readall to check each block for correct data (Ø's).
- D. After all blocks have been verified forward, the tape is reversed again and alternately searches for each block number in reverse sequence then writes each block to Ø's in Write Data mode. This insures parity conservation if the tape is to be read before being rewritten.
- E. The program then halts at address 2316g. To format another tape, perform the operations in section 4.Ø then press CONTINUE.

7.Ø ERRORS

7.1 ERROR HALT

Address 3Ø12g is the common error halt. Press CONTINUE to proceed from it.



**DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASSACHUSETTS 01754**