

[illegible][illegible]

```

LL      CCCCCCCC DDDDDDDD RRRRRRRR      I I I I I      VV      VV      EEEEEEEEEEE RRRRRRRR
LL      CCCCCCCC DDDDDDDD RRRRRRRR      I I I I I      VV      VV      EEEEEEEEEEE RRRRRRRR
LL      CC        DD        RR          II      VV      VV      EE          RR
LL      CC        DD        RR          II      VV      VV      EE          RR
LL      CC        DD        RR          II      VV      VV      EE          RR
LL      CC        DD        RR          II      VV      VV      EE          RR
LL      CC        DD        RRRRRRRR      II      VV      VV      EE          RR
LL      CC        DD        RRRRRRRR      II      VV      VV      EEEEEEEEEEE RRRRRRRR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LL      CC        DD        RR  RR       II      VV      VV      EE          RR
LLLLLLLLLLLL CCCCCCCC DDDDDDDD RR      RR      I I I I I      VV      VV      EEEEEEEEEEE RRRRRRRR
LLLLLLLLLLLL CCCCCCCC DDDDDDDD RR      RR      I I I I I      VV      VV      EEEEEEEEEEE RRRRRRRR

LL      I I I I I      SSSSSSSS
LL      I I I I I      SSSSSSSS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SSSSSS
LL      II          SSSSSS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SS
LL      II          SSSSSSSS
LLLLLLLLLLLL I I I I I      SSSSSSSS
LLLLLLLLLLLL I I I I I      SSSSSSSS

```

|     |     |                                                          |
|-----|-----|----------------------------------------------------------|
| (1) | 72  | Declarations                                             |
| (1) | 205 | Driver prologue table and driver dispatch table          |
| (1) | 254 | Function decision table                                  |
| (1) | 279 | FDT - Write function processing                          |
| (1) | 368 | Line printer driver                                      |
| (1) | 649 | Line printer interrupt dispatcher                        |
| (1) | 681 | Line printer unit initialization                         |
| (1) | 728 | Line printer unit initialization                         |
| (1) | 757 | Register dump routine                                    |
| (1) | 785 | FALLBACK - Macros to create character table for fallback |



```
0000 1
0000 2
0000 3
0000 4
0000 5
0000 6
0000 7
0000 8
0000 9
0000 10
0000 11
0000 12
0000 13
0000 14
0000 15
0000 16
0000 17
0000 18
0000 19
0000 20
0000 21
0000 22
0000 23
0000 24
0000 25
0000 26
0000 27
0000 28
0000 29
0000 30
0000 31
0000 32
0000 33
0000 34
0000 35
0000 36
0000 37
0000 38
0000 39
0000 40
0000 41
0000 42
0000 43
0000 44
0000 45
0000 46
0000 47
0000 48
0000 49
0000 50
0000 51
0000 52
0000 53
0000 54
0000 55
0000 56
0000 57
```

.TITLE LCDRIVER - LINE PRINTER DRIVER FOR LP-730  
.IDENT 'V04-000'

\*\*\*\*\*  
\*  
\* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY \*  
\* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. \*  
\* ALL RIGHTS RESERVED. \*  
\*  
\* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED \*  
\* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE \*  
\* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER \*  
\* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY \*  
\* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY \*  
\* TRANSFERRED. \*  
\*  
\* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE \*  
\* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT \*  
\* CORPORATION. \*  
\*  
\* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS \*  
\* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. \*  
\*  
\*\*\*\*\*

ABSTRACT:  
LINE PRINTER DRIVER FOR LP-730 CONTROLLER

AUTHOR:  
E. A. DRAYTON 4-NOV-80

MODIFIED BY:

V03-008 EMD0091 Ellen M. Dusseault 30-Apr-1984  
Add DEV\$M\_NNM characteristic to DEVCHAR2 so that these  
devices will have the prefix "node\$".

V03-007 EAD0113 Elliott A. Drayton 9-Mar-1984  
Added code to restore CRB address to R0.

V03-006 EMD0048 Ellen M. Dusseault 20-Feb-1984  
Add new feature, fallback. The ability to convert  
multinational characters (8-bit) to a 7-bit equivalent  
representation if possible. Otherwise an underscore  
replaces the multinational character.

V03-005 WHM0001 Bill Matthews 19-Dec-1983  
Added code to support new IDB fields IDB\$B\_COMBO\_VECTOR  
and IDB\$B\_COMBO\_CSR\_OFFSET for determining the main CSR  
address and loading the soft vector for the combo device.

V03-004 EAD0004 Elliott A. Drayton 26-Jul-1983



0000 58 :  
0000 59 :  
0000 60 :  
0000 61 :  
0000 62 :  
0000 63 :  
0000 64 :  
0000 65 :  
0000 66 :  
0000 67 :  
0000 68 :  
0000 69 :  
0000 70 :--

Add code to restore R2 after call to RELMPR, also  
increase the number of preallocated map registers.

V03-003 EAD0003 Elliott A. Drayton 25-Apr-1983  
Changed code to cause indirect registers BUFADR thru  
LINES to be loaded during a write physical operation.  
Also changed DIPL to 21 and default lines per page  
to 66 and do not allow a zero for lines per page.

V03-002 KDM0002 Kathleen D. Morse 28-Jun-1982  
Added \$DYNDEF, \$DCDEF, and \$PRDEF.

```
0000 72      .SBTTL Declarations
0000 73
0000 74 :
0000 75 : MACRO LIBRARY CALLS
0000 76 :
0000 77
0000 78 $CRBDEF      : CRB offsets
0000 79 $DCDEF      : Define device types
0000 80 $DDBDEF      : DDB offsets
0000 81 $DPTDEF      : DPT offsets
0000 82 $DYNDDEF     : Dynamic data structure types
0000 83 $EMBDEF      : Error msg buffer
0000 84 $IDBDEF      : IDB offsets
0000 85 $IODEF       : I/O function codes
0000 86 $IRPDEF      : IRP offsets
0000 87 $JIBDEF      : JIB offsets
0000 88 $LPDEF       : Line printer characteristics
0000 89 $MSGDEF      : System message types
0000 90 $PCBDEF      : Process Control Block
0000 91 $PRDEF       : Processor register numbers
0000 92 $SSDEF       : System status codes
0000 93 $UCBDEF      : UCB offsets
0000 94 $VECDDEF     : VEC offsets
0000 95 $VADEF       : Virtual address fields
0000 96 :
0000 97 : LOCAL SYMBOLS
0000 98 :
0000 99 : ARGUMENT LIST OFFSET DEFINITIONS
0000 100 :
0000 101
00000000 0000 102 P1=0      : First function dependent parameter
00000004 0000 103 P2=4      : Second function dependent parameter
00000008 0000 104 P3=8      : Third function dependent parameter
0000000C 0000 105 P4=12     : Fourth function dependent parameter
00000010 0000 106 P5=16     : Fifth function dependent parameter
00000014 0000 107 P6=20     : Sixth function dependent parameter
0000 108
0000 109 :
0000 110 : CHARACTER CODE DEFINITIONS
0000 111 :
0000 112
FFFFFEC 0000 113 DMF_CSR    = -20      : DMF CSR offset from LP CSR
00000002 0000 114 RESET     = 2        : Value for LP master reset
00000204 0000 115 M_STDCSR  = <2 * 256> + 4 : Set indirect reg. addr. to 2 and format on
0000 116
00000400 0000 117 C_INDREG4  = 4 * 256    : Address of indirect reg. number 5
00000006 0000 118 C_REGCNT  = 6        : Number of indirect registers to init
0000000F 0000 119 C_TIM     = 15       : Initial value for LPTIMEOUT
0000 120
00000005 0000 121 LC_MAPREG  = 5        : Number of map reg. to alloc
00000780 0000 122 LP_HRCNT  = 1920     : Timeout value for one hour
00000009 0000 123 LP_NUM_REGS = 9        : Total number of controller indirect reg.
00000200 0000 124 PGSIZ     = 512     : Memory page size(bytes)
00000042 0000 125 LINPAG    = 66      : Default lines per page for printer paper
0000 126
00000076 0000 127 LP_ERRBUF_LEN = <LP_NUM_REGS*4>+<<EMBSL_DV_REGSAV>+4> : Size of errorlog buffer
0000 128
```



LCDRIVER  
V04-000

- LINE PRINTER DRIVER FOR LP-730 K 5  
Declarations

16-SEP-1984 00:00:36 VAX/VMS Macro V04-00  
5-SEP-1984 00:14:49 [DRIVER.SRC]LCDRIVER.MAR;1

Page 4  
(1)

```
0000 130 :  
0000 131 : Local macros  
0000 132 :  
0000 133 :  
0000 134 .MACRO LOADUBAA  
0000 135 JSB G^IOC$LOADUBAMAPA  
0000 136 .ENDM LOADUBAA  
0000 137
```

```
0000 139 :  
0000 140 : DEVICE REGISTER OFFSET DEFINITIONS  
0000 141 :  
0000 142 :  
0000 143 $DEFINI LP  
0000 144 :  
0000 145 $DEF LP_CSR .BLKW 1 ;CONTROL STATUS REGISTER  
0002 146 $VIELD LP_CSR,0,<-  
0002 147 <PRIENA,,M>,- ; Print enable  
0002 148 <RESET,,M>,- ; Line printer controller reset  
0002 149 <FMTCTL,,M>,- ; Format enable  
0002 150 <UNUSED,2,M>,- ; Unused bits  
0002 151 <MAINT,,M>,- ; Maintance mode enable  
0002 152 <IE,,M>,- ; Interrupt enable  
0002 153 <RDY,,M>,- ; Line printer ready ( done )  
0002 154 <INDADD,3,M>,- ; Indirect register address  
0002 155 <UNUSED,,M>,- ; Unused bit  
0002 156 <CVERF,,M>,- ; Cable connected verification  
0002 157 <DAVERR,,M>,- ; Direct access vertical forms unit error  
0002 158 <LPERR,,M>,- ; Line printer error  
0002 159 <NEX,,M>,- ; Non existant memory error  
0002 160 >  
0002 161 :  
0002 162 $VIELD LP_CSR2,8,<-  
0002 163 <ADTOCR,,M>,- ; Automatic CR insert enable  
0002 164 <MECHFF,,M>,- ; Mech. form feed to multiple line feed enab  
0002 165 <NPC,,M>,- ; Nonprintable character enable  
0002 166 <RESERVED,,M>,- ; RESERVED  
0002 167 <WRAP,,M>,- ; Enable wrapping of lines > 132 characters  
0002 168 <UNUSED,2,M>,- ; Unused bits  
0002 169 <CASE,,M>,- ; Upper and lower case enable  
0002 170 >  
0002 171 :  
0002 172 $DEF LP_DBR .BLKW 1 ;DATA BUFFER REGISTER  
0004 173 :  
0004 174 $DEFEND LP  
0000 175 :  
00009000 0000 176 LP_CSR$M_ERR=<LP_CSR$M_NEX!LP_CSR$M_CVERF>;*** LP_CSR$M_DAVERR!***
```



```
0000 178 :  
0000 179 : DEFINE DEVICE DEPENDENT UNIT CONTROL BLOCK OFFSETS  
0000 180 :  
0000 181 :  
0000 182 $DEFINI UCB  
0000 183 :  
000000A0 0000 184 .=UCBSL-DPC+4  
00A0 185 $DEF UCB$W_CSR .BLKW 1 : Image of line printer CSR  
00A2 186 $DEF UCB$W_BYTXFR .BLKW 1 : Bytes transfered by controller  
00A4 187 $DEF UCB$W_LINCNT .BLKW 1 : Number of lines printed  
00A6 188 $DEF UCB$W_PREFIX .BLKW 1 : Prefix char and count(indirect reg. 2 )  
00A8 189 $DEF UCB$W_SUFFIX .BLKW 1 : Suffix char and count(indirect reg. 3 )  
00AA 190 $DEF UCB$W_BUFADR .BLKW 1 : Lower addr bits of buff addr(indirect reg.  
00AC 191 $DEF UCB$W_BYTECNT .BLKW 1 : DMA byte count (indirect reg. 5)  
00AE 192 $DEF UCB$B_HIBUFADR .BLKB 1 : Upper addr bits of buff addr(indirect reg.  
00AF 193 $DEF UCB$B_CSR2 .BLKB 1 : Control flags for LP attributes  
00B0 194 $DEF UCB$W_LINES .BLKW 1 : Number of lines per page (indirect reg. 7)  
00B2 195 $DEF UCB$W_LPTIMEOUT .BLKW 1 : Offline message timeout value  
00B4 196 $DEF UCB$W_LP_OFLCNT .BLKW 1 : Offline time counter  
00B6 197 $DEF UCB$W_MAXBYT .BLKW 1 : Number of map registers to alloc.  
00B8 198 $DEF UCB$S_REGSAVE .BLKL 2 : Save R1 and R2 here  
00C0 199 $DEF UCB$S_MAPREG .BLKL 1 : Save permanent map reg. info.  
00C4 200 :  
000000C4 00C4 201 UCB$K_SIZE=.  
00C4 202 :  
00C4 203 $DEFEND UCB
```

```
0000 205      .SBTTL Driver prologue table and driver dispatch table
0000 206
0000 207 :
0000 208 : LOCAL DATA
0000 209 :
0000 210 : DRIVER PROLOGUE TABLE
0000 211 :
0000 212
0000 213      DPTAB      -      ;DEFINE DRIVER PROLOGUE TABLE
0000 214      END=LP END,-      ; End of driver
0000 215      ADAPTER=UBA,-      ; Adapter type
0000 216      UCBSIZE=UCB$K_SIZE,-      ; UCB size
0000 217      NAME=LCDRIVER      ; Driver name
0038 218      DPT_STORE INIT      ; Control block init values
0038 219      DPT_STORE UCB,UCB$B_FIPL,B,8      ; Fork IPL
003C 220      DPT_STORE UCB,UCB$B_DEVCHAR,L,-      ; Device characteristics
003C 221      <DEV$M_REC-      ; Record oriented
003C 222      !DEV$M_AVL-      ; Available
003C 223      !DEV$M_CCL-      ; Carriage control device
003C 224      !DEV$M_ODV-      ; Output device
003C 225      !DEV$M_ELG-      ; Error logging
003C 226      >
0043 227      DPT_STORE UCB,UCB$B_DEVCHAR2,L,-      ; Device Characteristics
0043 228      <DEV$M_NNM>      ; prefix with "node$"
004A 229      DPT_STORE UCB,UCB$B_DEVCLASS,B,DC$ LP      ; Device class
004E 230      DPT_STORE UCB,UCB$B_DEVTYPE,B,DT$ DMF32      ; Device type
0052 231      DPT_STORE UCB,UCB$W_DEVBUFSIZ,W,132      ; Default buffer size
0057 232      DPT_STORE UCB,UCB$B_DEVDEPEND,L,<66a24+LP$M_MECHFORM>      ; Printer parameters
005E 233      DPT_STORE UCB,UCB$B_DIPL,B,21      ; Device IPL
0062 234      DPT_STORE REINIT      ; Control block re-init values
0062 235      DPT_STORE CRB,CRB$B_INTD+4,D,LP$INT      ; Interrupt service routine address
0067 236      DPT_STORE CRB,CRB$B_INTD+VEC$B_INITIAL,D,LC_CINIT      ; Controller init
006C 237      DPT_STORE CRB,CRB$B_INTD+VEC$B_UNITINIT,D,LC_INIT      ; Unit init
0071 238      DPT_STORE DDB,DDB$B_DDT,D,LP$DDT      ; DDT address
0076 239      DPT_STORE END      ;
0000 240
0000 241 :
0000 242 : DRIVER DISPATCH TABLE
0000 243 :
0000 244
0000 245      DDTAB      LP,-      ;DRIVER DISPATCH TABLE
0000 246      STARTIO,-      ; Start I/O operation
0000 247      0,-      ; Unsolicited interrupt
0000 248      FUNCTABLE,-      ; Function table
0000 249      +IOC$CANCELIO,-      ; Cancel I/O
0000 250      LP_REGDUMP,-      ; Register dump routine
0000 251      0,-      ; Size of diagnostic buffer
0000 252      LP_ERRBUF_LEN      ; Size of error log buffer
```



```
0038 254 .SBTTL Function decision table
0038 255 :
0038 256 : FUNCTION DECISION TABLE
0038 257 :
0038 258 :
0038 259 FUNCTABLE:
0038 260 FUNCTAB -
0038 261 <SENSECHAR,-
0038 262 SETCHAR,-
0038 263 SENSEMODE,-
0038 264 SETMODE,-
0038 265 WRITELBLK,-
0038 266 WRITEPBLK,-
0038 267 WRITEVBLK>
0040 268
0040 269 FUNCTAB
0048 270 FUNCTAB LP_WRITE,<WRITELBLK,-
0048 271 WRITEPBLK,-
0048 272 WRITEVBLK>
0054 273 FUNCTAB +EXESSETMODE,-
0054 274 <SETCHAR,SETMODE>
0060 275 FUNCTAB +EXESSENSEMODE,-
0060 276 <SENSECHAR,-
0060 277 SENSEMODE>

;FUNCTION DECISION TABLE
; Legal functions
; Sense characteristics
; Set characteristics
; Sense mode
; Set mode
; Write logical block
; Write physical block
; Write virtual block

;NO BUFFERED FUNCTIONS
; Write functions

; Set characteristics functions
; Sense characteristics
; Sense mode
```



```
006C 279 .SBTTL FDT - Write function processing
006C 280 :+
006C 281 : LP_WRITE - WRITE FUNCTION PROCESSING
006C 282 :
006C 283 : THIS ROUTINE IS CALLED FROM THE FUNCTION DECISION TABLE DISPATCHER TO PROCESS
006C 284 : A WRITE PHYSICAL, WRITE LOGICAL, OR WRITE VIRTUAL FUNCTION TO A LINE PRINTER.
006C 285 :
006C 286 : INPUTS:
006C 287 :
006C 288 : R0 = SCRATCH.
006C 289 : R1 = SCRATCH.
006C 290 : R2 = SCRATCH.
006C 291 : R3 = ADDRESS OF I/O REQUEST PACKET.
006C 292 : R4 = CURRENT PROCESS PCB ADDRESS.
006C 293 : R5 = ASSIGNED DEVICE UCB ADDRESS.
006C 294 : R6 = ADDRESS OF CCB.
006C 295 : R7 = I/O FUNCTION CODE.
006C 296 : R8 = FUNCTION DECISION TABLE DISPATCH ADDRESS.
006C 297 : R9 = SCRATCH.
006C 298 : R10 = SCRATCH.
006C 299 : R11 = SCRATCH.
006C 300 : AP = ADDRESS OF FIRST FUNCTION DEPENDENT PARAMETER.
006C 301 :
006C 302 : OUTPUTS:
006C 303 :
006C 304 : FOR PROCESSING BY THE LINE PRINTER DRIVER.
006C 305 :-
006C 306
006C 307 LP_WRITE:
3C A3 0C AC DO 006C 308 MOVL P4(AP),IRPSB_CARCON(R3) ; Insert carriage control byte
51 04 AC 3C 0071 309 MOVZWL P2(AP),R1 ; Number of bytes to transfer
0E 13 0075 310 BEQL 10$ ; If EQL none
50 6C DO 0077 311 MOVL P1(AP),R0 ; Get starting virtual addr. of tran
OC 44 A5 09 EO 007A 312 BBS #LPSV_FALLBACK,UCBSL_DEVDEPEND(R5),20$ ; If set, fallback
00000000'GF 16 007F 313 5$: JSB G^EXE$WRITELOCK ; Check buffer and lock in memory
00000000'GF 17 0085 314 10$: JMP G^EXE$QIODRVPKT ; Queue I/O packet to driver
008B 315
008B 316
008B 317 : Convert IRP to a buffered i/o packet since fallback is specified.
008B 318 : Contingent upon the fact that passall is not set (since passall would take
008B 319 : precedence) and upon the fact that this is not a physical write operation
008B 320 : which again has precedence.
008B 321
EF 44 A5 08 EO 008B 322 20$: BBS #LPSV_PASSALL,UCBSL_DEVDEPEND(R5),5$ ; passall has precedence
0B 57 91 0090 323 CMPB R7,#10$_WRITEPBLK ; physical write
EA 13 0093 324 BEQL 5$ ; EQL, yes so don't convert
3A A4 B7 0095 325 DECW PCBSW_BIOCNT(R4) ; decrement buffered io count
59 19 0098 326 BLSS 30$ ; if negative, quota exceeded
3E A4 B6 009A 327 25$: INCW PCBSW_DIOCNT(R4) ; readjust direct io count
2A A3 01 A8 009D 328 BISW #IRPSM_BUFIO,IRPSW_STS(R3) ; set irp$sm_bufio equal to 1
03 BB 00A1 329 PUSH R ; save buffer address and length
00000000'GF 16 00A3 330 JSB G^EXE$WRITECHK ; do we have access to user data
51 0C C0 00A9 331 ADDL #12,R1 ; add 12 bytes for buffer header
53 DD 00AC 332 PUSH R3 ; save irp address
00000000'GF 16 00AE 333 JSB G^EXE$BUFFRQUOTA ; ensure user has sufficient resourc
55 50 E9 00B4 334 BLBC R0,40$ ; if clr, no
00000000'GF 16 00B7 335 JSB G^EXE$ALLOCBUF ; allocate system buffer
```



```
000003E1'FF 20 4C 50 E9 00BD 336 BLBC R0,40$ ; if clr, no buffer allocated
50 2C A3 53 8ED0 00C0 337 POPL R3 ; restore irp address
20 0080 52 D0 00C3 338 MOVL R2,IRPSL_SVAPTE(R3) ; save address of buffered i/o packe
30 A0 51 C2 00C7 339 MOVL PCBSL_JIB(R4),R0 ; get jib address
62 0C A2 03 BA 00D0 340 SUBL R1,JIBSL_BYTCNT(R0) ; adjust buffered i/o quota
04 A2 51 B0 00D4 341 MOVW R1,IRPSW_BOFF(R3) ; set number of bytes charged to quo
52 0C A2 9E 00D6 342 POPR #M<R0,R1> ; restore to user buf addr and lengt
38 BB 00DA 343 MOVAB 12(R2),(R2) ; set up system buffer header
20 60 50 D0 00DE 344 MOVL R0,4(R2) ; insert address of user data
62 51 9E 00DE 345 MOVAB 12(R2),R2 ; point at data area in buffer
38 2E 00E2 346 PUSH R1,(R0),#X20,@TRANS_TAB,R1,(R2) ; save registers that movtc destroys
92 11 00E4 347 MOVTC R1,(R0),#X20,@TRANS_TAB,R1,(R2) ; move char. into system buffer
38 BA 00EF 348 POPR #M<R3,R4,R5> ; restore registers
92 11 00F1 349 BRB 10$ ;
00F3 350
00F3 351
30$: 3A A4 B6 00F3 352 INCW PCBSW_BIOCNT(R4) ; Backout charge
52 3A A4 3E 00F6 353 MOVAV PCBSW_BIOCNT(R4),R2 ; store pcb buffer count address
00000000'GF 0B BB 00FA 354 PUSH R1,R3> ; save registers before destroyed
10 50 E9 00FC 355 JSB G^EXE$SNGLEQUOTA ; check unit quota of io function ty
3A A4 B7 0105 356 BLBC R0,50$ ; Clr, abort io
8E 11 0107 357 POPR #M<R0,R1,R3> ; restore registers
53 8ED0 010C 358 DECW PCBSW_BIOCNT(R4) ; charge for i/o of type
00000000'GF 17 010A 359 BRB 25$ ; all set, continue on
010C 360
40$: 53 8ED0 010C 361 POPL R3 ; restore irp address
00000000'GF 17 010F 362 JMP G^EXE$ABORTIO ;
0115 363
50$: 0E BA 0115 364 POPR #M<R1,R2,R3> ; restore registers
00000000'GF 17 0117 365 JMP G^EXE$ABORTIO ;
011D 366
```

```
011D 368 .SBTTL Line printer driver
011D 369 :+
011D 370 : STARTIO - START I/O OPERATION ON LINE PRINTERS
011D 371 :
011D 372 : THIS ROUTINE IS ENTERED WHEN THE ASSOCIATED UNIT IS IDLE AND A PACKET
011D 373 : IS AVAILABLE.
011D 374 :
011D 375 : INPUTS:
011D 376 :
011D 377 : R3 = ADDRESS OF I/O REQUEST PACKET.
011D 378 : R5 = UCB ADDRESS FOR IDLE UNIT.
011D 379 :
011D 380 : OUTPUTS:
011D 381 :
011D 382 : NO EXPLICIT OUTPUTS - THE UNIT IS IN WAITING FOR INTERRUPT STATE
011D 383 : OR THE I/O IS COMPLETE.
011D 384 :-
011D 385 :
011D 386 STARTIO:
011D 387 CLRW UCB$W_CSR(R5) ; Clr storage area
0121 388 :
0121 389 : Retrieve the address of the device CSR
0121 390 :
54 24 A5 D0 0121 391 MOVL UCB$C_CRB(R5),R4 ; Get address of CRB
54 2C B4 D0 0125 392 MOVL @CRB$C_INTD+VEC$C_IDB(R4),R4 ; Get device CSR address
0129 393 :
0129 394 : What is the function?
0129 395 :
00 ED 0129 396 CMPZV #IRPSV_FCODE,- ; WRITELBLK function?
06 012B 397 #IRPSS_FCODE,-
20 A3 012C 398 IRPSW_FUNC(R3),-
20 012E 399 #IOS_WRITELBLK
03 12 012F 400 BNEQ 10$ ; If NEQ, some other function
0044 31 0131 401 BRW WRITE ; It is EQL, it's a write function
00 0134 402 :
06 ED 0134 403 10$: CMPZV #IRPSV_FCODE,- ; Write physical block function?
06 0136 404 #IRPSS_FCODE,-
20 A3 0137 405 IRPSW_FUNC(R3),-
0B 0139 406 #IOS_WRITEPBLK
03 12 013A 407 BNEQ 15$ ; If NEQ no.
0040 31 013C 408 BRW WRITEP ; Else go do it.
00 013F 409 :
06 ED 013F 410 15$: CMPZV #IRPSV_FCODE,- ; Set mode function?
20 A3 0141 411 #IRPSS_FCODE,-
23 0142 412 IRPSW_FUNC(R3),-
05 13 0144 413 #IOS_SETMODE
0145 414 BEQL 30$ ; If EQL yes.
0147 415
```



```
0147 417
0147 418 :
0147 419 : THE SET MODE AND SET CHARACTERISTICS INFORMATION IS NOT MOVED
0147 420 : TO THE BOARDS REGISTERS UNTIL IO IS STARTED.
0147 421 :
0147 422 :
0147 423 :
0147 424 : Set characteristics
0147 425 :
0147 426 :
38 A3 B0 0147 427 20$: MOVW IRP$L_MEDIA(R3),-
40 A5 014A 428 UCB$B_DEVCLASS(R5) ;
014C 429
014C 430 :
014C 431 : Set mode
014C 432 :
014C 433 :
3A A3 90 014C 434 30$: MOVB IRP$L_MEDIA+2(R3),- ; Set Page Width - DMF only has a BYTE
42 A5 014F 435 UCB$W_DEVBUFSIZ(R5)
0151 436
50 14 3C 0151 437 MOVZWL #SS$_BADPARAM,R0 ; Set BADPARAM error status
3F A3 95 0154 438 TSTB IRP$L_MEDIA+7(R3) ; Lines per page EQL zero
1A 13 0157 439 BEQL 35$ ; Yes, then branch
0159 440
3C A3 D0 0159 441 MOVL IRP$L_MEDIA+4(R3),- ; Set device dependent flags
44 A5 015C 442 UCB$L_DEVDEPEND(R5)
015E 443
3F A3 90 015E 444 MOVB IRP$L_MEDIA+7(R3),- ; Set lines per page
00B0 C5 0161 445 UCB$W_LINES(R5)
0164 446
3A A3 90 0164 447 MOVB IRP$L_MEDIA+2(R3),- ; Set Page Width BYTE for DMF32
00B1 C5 0167 448 UCB$W_LINES+1(R5)
016A 449
44 A5 90 016A 450 MOVB UCB$L_DEVDEPEND(R5),- ; Set device dependant controller bits
00AF C5 016D 451 UCB$B_CSR2(R5)
0170 452
50 01 3C 0170 453 MOVZWL #SS$_NORMAL,R0 ; Set normal return status.
51 D4 0173 454 35$: CLRL R1
0175 455
0147 31 0175 456 BRW EXIT
```

```
0178 458 ;  
0178 459 ; Is PASSALL set?  
0178 460 ;  
0178 461 ;  
44 08 E0 0178 462 WRITE: BBS #LPSV_PASSALL,- ; If set do physical I/O.  
A5 017A 463 UCBSL_DEVDEPEND(R5),-  
02 017C 464 WRITEP  
OF 11 017D 465 BRB DECODE ; Else decode carriage control info  
017E 466  
017F 467 .ENABL LSB  
017F 468 ;  
017F 469 ; Perform write physical block  
017F 470 ;  
017F 471 ;  
017F 472 WRITEP:  
0400 8F B0 017F 473 MOVW #C_INDREG4,- ; Load the controller CSR  
64 0183 474 LP_CSR(R4) ; ...and set indirect register address  
51 00AA C5 3E 0184 475 MOVAW UCBSW_BUFADR(R5),R1 ; Address of indirect reg. data  
52 04 D0 0189 476 MOVL #C_REGCNT-2,R2 ; Number of indirect registers to copy  
19 11 018C 477 BRB COMMON ; Ok, do what everyone else would  
018E 478  
018E 479 ;  
018E 480 ; Decode carriage control info.  
018E 481 ;  
018E 482 DECODE:  
00000000'GF 16 018E 483 JSB G^EXESCARRIAGE ; Returns carr. control in IRPSB_CARCON  
0194 484 ASSUME UCBSW_PREFIX+2 EQ UCBSW_SUFFIX  
3C A3 D0 0194 485 MOVL IRPSB_CARCON(R3),- ; Set prefix char and cnt in cont. image  
00A6 C5 0197 486 UCBSW_PREFIX(R5)  
64 0204 8F B0 019A 487 MOVW #M_STDCSR,LP_CSR(R4) ; Setup for typical transfer  
52 06 D0 019F 488 MOVL #C_REGCNT,R2 ; Number of reg. to init. in controller  
51 00A6 C5 3E 01A2 489 MOVAW UCBSW_PREFIX(R5),R1 ; Get address of reg. images
```



```
01A7 491 COMMON:
01A7 492
7E A5 B5 01A7 493 TSTW UCB$W_BCNT(R5) ; Is this a zero length transfer?
0A 12 01AA 494 BNEQ 10$ ; If NEQ, no
00AE C5 94 01AC 495 CLR B UCB$B_HIBUFADR(R5) ; High bits of buffer address
00AC C5 B4 01B0 496 CLRW UCB$W_BYTECNT(R5) ; Byte count
74 11 01B4 497 BRB 30$
01B6 498 ;
01B6 499 ; Allocate UBA map registers
01B6 500 ;
01B6 501
01B6 502 10$:
01B6 503
00B8 C5 51 7D 01B6 504 MOVQ R1,UCB$L_REGSAVE(R5) ; Save registers R1 and R2
00B6 C5 B1 01BB 505 CMPW UCB$W_MAXBYT(R5),- ; If bcnt is > max bytes then get more map r
7E A5 01BF 506 UCB$W_BCNT(R5)
16 18 01C1 507 BGEQ 20$ ; ...else load UBA map registers
52 24 A5 D0 01C3 508 MOVL UCB$L_CRB(R5),R2 ; Get the CRB address
34 A2 D0 01C7 509 MOVL CRB$L_INTD+VEC$W_MAPREG(R2),- ; Save pre alloc. map register info
00C0 C5 01CA 510 UCB$L_MAPREG(R5)
01CD 511
8000 8F AA 01CD 512 BICW #VEC$M_MAPLOCK,- ; Clear map register lock bit
34 A2 01D1 513 CRB$L_INTD+VEC$W_MAPREG(R2)
01D3 514
01D3 515 REQMPR ; Request UNIBUS map registers
01D9 516 20$:
01D9 517 ;
01D9 518 ; If buffered flag set, irp is buffered io which first must be
01D9 519 ; converted to a direct io before proceeding further.
01D9 520 ;
1E 2A A3 00 E1 01D9 521 BBC #IRP$V_BUFIO,IRP$W_STS(R3),25$ ; IF Set, buffered io
52 50 2C A3 D0 01DE 522 MOVL IRP$L_SVAPTE(R3), R0 ; Get system buffer address
52 60 15 09 EF 01E2 523 EXTZV #VAS$V_VPN,#VASS_VPN,(R0),R2 ; Extract vpn from system buffer add
51 00000000 GF D0 01E7 524 MOVL G*MMG$GL_SPTBASE,R1 ; Get address of base of system page
78 A5 6142 DE 01EE 525 MOVAL (R1)[R2],UCB$L_SVAPTE(R5) ; Store address of pte in ucb field
52 60 09 00 EF 01F3 526 EXTZV #VAS$V_BYTE,#VASS_BYTE,(R0),R2 ; Byte offset in system page
7C A5 52 B0 01F8 527 MOVW R2,UCB$W_BOFF(R5) ; Store this offset in ucb field
01FC 528 25$:
01FC 529 LOADUBAA ; load uba map registers
0202 530 ;
0202 531 ; Load byte count and calculate UNIBUS transfer address
0202 532 ;
0202 533
7E A5 AE 0202 534 MNEGW UCB$W_BCNT(R5),- ; Setup byte count
00AC C5 0205 535 UCB$W_BYTECNT(R5)
51 7C A5 3C 0208 536 MOVZWL UCB$W_BOFF(R5),R1 ; Fetch byte offset
52 24 A5 D0 020C 537 MOVL UCB$L_CRB(R5),R2 ; Get CRB address
34 A2 F0 0210 538 INSV CRB$L_INTD+VEC$W_MAPREG(R2),-
51 09 09 0213 539 #9,#9,R1
52 51 02 10 EF 0216 540 EXTZV #16,#2,R1,R2 ;Get memory extension bits
00AA C5 51 B0 021B 541 MOVW R1,UCB$W_BUFADR(R5) ;Setup 16 bit buffer address
00AE C5 52 90 0220 542 MOV B R2,UCB$B_HIBUFADR(R5) ; Set upper address bits
51 00B8 C5 7D 0225 543 MOVQ UCB$L_REGSAVE(R5),R1 ;Restore registers R1 and R2
022A 544
022A 545 ;
022A 546 ; Load controller and start transfer
022A 547 ;
```

|      |    |    |    |      |     |       |        |                                                     |
|------|----|----|----|------|-----|-------|--------|-----------------------------------------------------|
| 02   | A4 | 81 | B0 | 022A | 548 |       |        |                                                     |
|      | F9 | 52 | F5 | 022A | 549 | 30\$: | MOVW   | (R1)+,LP_DBR(R4) ; Copy indirect reg. to controller |
|      |    |    |    | 022E | 550 |       | SOBGTR | R2,30\$ ; Branch if more to do.                     |
|      |    |    |    | 0231 | 551 |       |        |                                                     |
|      |    |    |    | 0231 | 552 | 40\$: | DSBINT | ; Raise IPL                                         |
|      | 05 |    | E0 | 0237 | 553 |       | BBS    | #UCBSV POWER,- ; If set, power has failed.          |
| 20   | 64 | A5 |    | 0239 | 554 |       |        | UCBSW STS(R5),41\$                                  |
| 0041 | 8F |    | A8 | 023C | 555 |       | BISW   | #LP_CSRSM_PRIENA!LP_CSRSM_IE,-                      |
|      | 64 |    |    | 0240 | 556 |       |        | LP_CSR(R4) ; Start xfer                             |
|      |    |    |    | 0241 | 557 |       | WFIKPC | 140\$,#12 ; Wait for xfer to complete               |
|      |    |    |    | 024B | 558 |       | IOFORK | ; Fork to lower IPL                                 |



```
0251 560 :  
0251 561 : Save status and check for errors  
0251 562 :  
0251 563 :  
00A0 C5 64 B0 0251 564 MOVW LP_CSR(R4),UCBSW_CSR(R5); Save controller state  
0E E0 0256 565 BBS #LP_CSR$V_LPERR,- ; If set printer problem  
6F 00A0 C5 0258 566 UCB$W_CSR(R5),140$  
64 B4 025C 567 41$: CLRW LP_CSR(R4) ; Set indirect register addr. to zero  
00A2 C5 02 A4 B0 025E 568 MOVW LP_DBR(R4),UCBSW_BYTXFR(R5) ; Save the number of bytes transfered  
00A4 C5 02 A4 B0 0264 569 MOVW LP_DBR(R4),UCBSW_LINCNT(R5) ; Save the number of lines printed  
026A 570 PURDPR ; Purge the data paths  
52 24 A5 D0 0270 571 MOVL UCB$L_CRB(R5),R2 ; Get the address of the CRB  
0F E0 0274 572 BBS #VEC$V_MAPLOCK,- ; If SET do not release map registers  
34 A2 0276 573 CRB$L_INTD+VEC$W_MAPREG(R2),-  
10 0278 574 42$  
0279 575 RELMPR ; Release map registers  
52 24 A5 D0 027F 576 MOVL UCB$L_CRB(R5),R2 ; Restore the address of the CRB  
00C0 C5 D0 0283 577 MOVL UCB$L_MAPREG(R5),- ; Restore pre-allocated map information  
34 A2 0287 578 CRB$L_INTD+VEC$W_MAPREG(R2)  
0289 579 42$: BBSC #UCBS$V_POWER,- ; If set, power has failed.  
05 E4 0289 580 UCB$W_STS(R5),90$  
37 64 A5 028B 581  
028E 582  
00A0 C5 9000 8F B3 028E 583 BITW #LP_CSR$M_ERR,UCBSW_CSR(R5) ; Any errors?  
09 13 0295 584 BEQL 44$ ; If EQL, no errors.  
00000000 GF 16 0297 585 JSB G^ERL$DEVICERR ; Report the error  
0006 31 029D 586 BRW 46$  
02A0 587  
64 A5 08 B3 02A0 588 44$: BITW #UCBS$M_CANCEL,UCBSW_STS(R5) ; Is the cancel bit set?  
06 13 02A4 589 BEQL 48$  
02A6 590  
50 2C 3C 02A6 591 46$: MOVZWL #SS$ _ABORT,R0 ; Set operation aborted status  
0003 31 02A9 592 BRW 50$  
02AC 593  
02AC 594  
02AC 595 :  
02AC 596 : I/O OPERATION SUCCESSFULLY COMPLETED  
02AC 597 :  
02AC 598  
50 01 3C 02AC 599 48$: MOVZWL #SS$ _NORMAL,R0 ; Set normal completion status  
02AF 600 50$:  
53 58 A5 D0 02AF 601 MOVL UCB$L_IRP(R5),R3 ; Retrieve address of i/o packet  
51 00A4 C5 3C 02B3 602 MOVZWL UCB$W_LINCNT(R5),R1 ; Get number of lines printed  
00A2 C5 F0 02B8 603 INSV UCB$W_BYTXFR(R5),-  
50 10 10 02BC 604 #16,#T6,R0 ; Insert number of characters in status  
02BF 605 EXIT:  
02BF 606 60$: REQCOM ; COMPLETE I/O REQUEST  
02C5 607  
02C5 608 90$: ENBINT  
FE52 31 02C8 609 BRW STARTIO
```

```
02CB 611 :  
02CB 612 : PRINTER IS NOT READY OR HAS PAPER PROBLEM  
02CB 613 :  
02CB 614 :  
64 00B4 C5 B4 02CB 615 140$: CLRW UCBSW_LP_OFLCNT(R5) ; Clear offline counter  
0040 8F AA 02CF 616 BICW #LP_CSRM_IE_LP_CSR(R4) ; Disable printer interrupt  
64 4000 8F B3 02D4 617 150$: SETIPL UCBSB_FIPC(R5) ; Lower to fork level  
08 12 02D8 618 BITW #LP_CSRM_LPERR_LP_CSR(R4) ; Printer still have paper problem?  
00B2 C5 0F B0 02DD 619 BNEQ 155$ ; If NEQ, yes it does  
FF4A 31 02DF 620 MOVW #C_TIM,UCBSW_LPTIMEOUT(R5) ; Reset timeout value  
02E4 621 BRW 40$ ; And continue transfer.  
02E7 622  
64 A5 08 B3 02E7 623 155$: BITW #UCBSM_CANCEL,UCBSW_STS(R5) ; Is the cancel bit set?  
03 13 02EB 624 BEQL 156$ ; If EQL, no cancel  
FF6C 31 02ED 625 BRW 41$  
02F0 626  
01 00B2 C5 3D 02F0 627 156$: ACBW UCBSW_LPTIMEOUT(R5),#1,-  
0026 00B4 C5 02F5 628 UCBSW_LP_OFLCNT(R5),160$; Skip until timeout  
02FA 629  
00B4 C5 B4 02FA 630 CLRW UCBSW_LP_OFLCNT(R5) ; Reset counter  
02FE 631  
0780 8F B1 02FE 632 CMPW #LP_HRCNT,- ; Are we at our max. (1 HR.)  
00B2 C5 0302 633 UCBSW_LPTIMEOUT(R5)  
0305 634  
00B2 C5 05 1B 0305 635 BLEQU 157$ ; If LEQ, yes  
02 A4 0307 636 MULW2 #2,UCBSW_LPTIMEOUT(R5) ; Double message timeout value  
18 BB 030C 637 157$: PUSHR #*M<R3,R4> ; Save registers  
54 05 9A 030E 638 MOVZBL #MSG$_DEVOFFLIN,R4 ; Set up message type  
53 00000000 GF 9E 0311 639 MOVAB G^SYS$GL_OPRMBX,R3 ; Address target mailbox  
00000000 GF 16 0318 640 JSB G^EXESSNDEVMSG ; Send message ignore error  
18 BA 031E 641 POPR #*M<R3,R4> ; Restore registers  
0320 642 160$: DSBINT UCBSB_DIPL(R5) ; Disable interrupts  
0327 643 WFIKPC 150$,#2 ; Wait for an interrupt or timeout  
0331 644 IOFORK ; Create for process  
9B 11 0337 645 BRB 150$  
0339 646 .DSABL LSB  
0339 647
```



```
0339 649 .SBTTL Line printer interrupt dispatcher
0339 650 :+
0339 651 : LPSINT - LINE PRINTER INTERRUPT DISPATCHER.
0339 652 :
0339 653 : THIS ROUTINE IS ENTERED VIA A JSB INSTRUCTION WHEN AN INTERRUPT OCCURS ON
0339 654 : THE LINE PRINTER CONTROLLER. THE STATE OF THE STACK ON ENTRY IS:
0339 655 :
0339 656 : 00(SP) = ADDRESS OF IDB ADDRESS.
0339 657 : 04(SP) = SAVED R3.
0339 658 : 08(SP) = SAVED R4.
0339 659 : 12(SP) = SAVED R5.
0339 660 : 16(SP) = INTERRUPT PC.
0339 661 : 20(SP) = INTERRUPT PSL.
0339 662 :
0339 663 : INTERRUPT DISPATCHING OCCURS AS FOLLOWS:
0339 664 :
0339 665 : IF THE INTERRUPT IS EXPECTED, THEN THE DRIVER IS CALLED AT ITS INTERRUPT
0339 666 : WAIT ADDRESS. ELSE THE INTERRUPT IS DISMISSED.
0339 667 :-
0339 668 :
0339 669 LPSINT::
OC 64 53 9E D0 0339 670 MOVL @ (SP)+, R3 ; Entry from dispatch
54 63 7D 033C 671 MOVQ IDB$L CSR(R3), R4 ; Get address of IDB
A5 01 E5 033F 672 BBCC #UCB$V INT, UCB$W STS(R5), 10$ ; If clr, interrupt not expected
64 0040 8F AA 0344 673 BICW #LP CSR$M IE, LP CSR(R4) ; Disable output interrupts
53 10 A5 D0 0349 674 MOVL UCB$L FR3(R5), R3 ; Restore remainder of driver context
OC B5 16 034D 675 JSB @UCB$C FPC(R5) ; Call driver at interrupt wait address
50 8E 7D 0350 676 10$: MOVQ (SP)+, R0 ; Restore registers
52 8E 7D 0353 677 MOVQ (SP)+, R2
54 8E 7D 0356 678 MOVQ (SP)+, R4
02 0359 679 REI
```

```
035A 681 .SBTTL Line printer unit initialization
035A 682
035A 683 :+ LC_INIT - LINE PRINTER UNIT INITIALIZATION
035A 684
035A 685 : THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER FAILURE. THE
035A 686 : ONLINE BIT IS SET FOR THE SPECIFIED UNIT.
035A 687
035A 688 : INPUTS:
035A 689
035A 690 : R5 = ADDRESS OF DEVICE UCB.
035A 691
035A 692 : OUTPUTS:
035A 693
035A 694 : THE ONLINE BIT IS SET IN THE DEVICE UCB AND THE ADDRESS OF THE UCB
035A 695 : IS FILLED INTO THE IDB OWNER FIELD.
035A 696 :-
035A 697
035A 698 LC_INIT: ; LINE PRINTER UNIT INITIALIZATION
035A 699
035A 700 BISW #UCB$M_ONLINE,- ; Set unit online
035C 701 UCB$W_STS(R5)
035E 702
035E 703 BBS #UCB$V_POWER,- ; If set power has failed
0360 704 UCB$W_STS(R5),10$
0363 705
0363 706 MOVL #LC_MAPREG,R3 ; Number of mapping reg. to pre-alloc
0366 707 JSB G*IDCSALOUBAMAPN ; Request map registers
036C 708 BLBC R0,10$ ; If LBC, request failed
036F 709 SUBW3 #2,R3,UCB$W_MAXBYT(R5) ; One less reg. for buf. near page boundry
0375 710 MULW2 #PAGSIZ,UCB$W_MAXBYT(R5) ; Save max byte count
037C 711 MOVL UCB$L_CRB(R5),R0 ; Get address of CRB
0380 712 BISW #VEC$M_MAPLOCK,- ; Set maplock bit
0384 713 CRB$L_INTD+VEC$W_MAPREG(R0)
0386 714 BRW 20$
0389 715
0389 716 10$: MOVL UCB$L_CRB(R5),R0 ; Get address of CRB
038D 717 CLRW UCB$W_MAXBYT(R5) ; Clear max byte cnt. since no map alloc
0391 718
0391 719 20$: MOVB UCB$W_DEVBUFSIZ(R5),- ; Set default buffer size
0394 720 UCB$W_LINES+1(R5)
0397 721
0397 722 BISB #LPSM_LOWER,UCB$B_CSR2(R5) ; Set device dependant bits
039D 723 MOVB #LINPAG,UCB$W_LINES(R5) ; Set lines per page
03A3 724 MOVL CRB$L_INTD+VEC$L_IDB(R0),R0 ; Get address of IDB
03A7 725 MOVL R5,IDB$L_OWNER(R0) ; Set address of device UCB
03AB 726 RSB
```

10 AB 035A 700 BISW #UCB\$M\_ONLINE,- ; Set unit online  
64 A5 035C 701 UCB\$W\_STS(R5)  
05 E0 035E 702  
26 64 A5 035E 703 BBS #UCB\$V\_POWER,- ; If set power has failed  
0360 704 UCB\$W\_STS(R5),10\$  
53 05 D0 0363 705  
00000000 GF 16 0363 706 MOVL #LC\_MAPREG,R3 ; Number of mapping reg. to pre-alloc  
1A 50 E9 0366 707 JSB G\*IDCSALOUBAMAPN ; Request map registers  
00B6 C5 53 02 A3 036C 708 BLBC R0,10\$ ; If LBC, request failed  
00B6 C5 0200 8F A4 036F 709 SUBW3 #2,R3,UCB\$W\_MAXBYT(R5) ; One less reg. for buf. near page boundry  
50 24 A5 D0 0375 710 MULW2 #PAGSIZ,UCB\$W\_MAXBYT(R5) ; Save max byte count  
8000 8F A8 037C 711 MOVL UCB\$L\_CRB(R5),R0 ; Get address of CRB  
34 A0 0380 712 BISW #VEC\$M\_MAPLOCK,- ; Set maplock bit  
0008 31 0384 713 CRB\$L\_INTD+VEC\$W\_MAPREG(R0)  
50 24 A5 D0 0386 714 BRW 20\$  
00B6 C5 B4 0389 715  
42 A5 90 0389 716 10\$: MOVL UCB\$L\_CRB(R5),R0 ; Get address of CRB  
00B1 C5 B4 038D 717 CLRW UCB\$W\_MAXBYT(R5) ; Clear max byte cnt. since no map alloc  
0391 718  
42 A5 90 0391 719 20\$: MOVB UCB\$W\_DEVBUFSIZ(R5),- ; Set default buffer size  
00B1 C5 0394 720 UCB\$W\_LINES+1(R5)  
0397 721  
00AF C5 80 8F 88 0397 722 BISB #LPSM\_LOWER,UCB\$B\_CSR2(R5) ; Set device dependant bits  
00B0 C5 42 8F 90 039D 723 MOVB #LINPAG,UCB\$W\_LINES(R5) ; Set lines per page  
50 2C A0 D0 03A3 724 MOVL CRB\$L\_INTD+VEC\$L\_IDB(R0),R0 ; Get address of IDB  
04 A0 55 D0 03A7 725 MOVL R5,IDB\$L\_OWNER(R0) ; Set address of device UCB  
05 03AB 726 RSB



```
03AC 728 .SBTTL Line printer unit initialization
03AC 729 :+
03AC 730 : LC_CINIT - LINE PRINTER CONTROLLER INITIALIZATION
03AC 731 :
03AC 732 : THIS ROUTINE IS CALLED AT SYSTEM STARTUP AND AFTER A POWER FAILURE. THE
03AC 733 : ONLINE BIT IS SET FOR THE SPECIFIED UNIT.
03AC 734 :
03AC 735 : INPUTS:
03AC 736 :
03AC 737 : R4 = ADDRESS OF DEVICE CSR.
03AC 738 : R5 = ADDRESS OF DEVICE UCB.
03AC 739 :
03AC 740 : OUTPUTS:
03AC 741 :
03AC 742 : THE LINE PRINTER OFFLINE MESSAGE COUNT IS INITIALIZED AFTER A
03AC 743 : RESET HAS BEEN DONE ON THE LP CONTROLLER AND THE INTERRUPT
03AC 744 : VECTOR OF THE LP IS LOADED INTO THE CONTROLLER.
03AC 745 :-
03AC 746 :
03AC 747 LC_CINIT:
03AC 748 MOVW #RESET,LP_CSR(R4) ; Master reset
03AF 749 MOVL IDBSL_UCB[ST(R5),R0 ; Get UCB address
03B3 750 MOVW #C_TIM,UCBSW_LPTIMEOUT(R0) ; Initialize LP offline message time
03B8 751 CVTBL IDBSB_COMBO_CSR_OFFSET(R5),R0 ; GET OFFSET TO MAIN DMF CSR
03BC 752 SUBB3 IDBSB_COMBO_VECTOR_OFFSET(R5),-
03BF 753 IDBSB_VECTOR(R5),(R4)[R0] ; LOAD THE VECTOR ADDRESS
03C3 754 MOVAL FALLTAB,TRANS_TAB ; Get address of fallback tables
03CE 755 RSB ;
```

000003E1'EF 000003E5'EF

50 64 02 B0  
00B2 C0 18 A5 D0  
50 0F A5 B0  
6440 0B A5 98  
000003E1'EF 000003E5'EF 83  
DE 03C3  
05 03CE

```
03CF 757 .SBTTL Register dump routine
03CF 758 :++
03CF 759 : LP_REGDUMP - REGISTER DUMP ROUTINE
03CF 760 :
03CF 761 : FUNCTIONAL DESCRIPTION:
03CF 762 :
03CF 763 : THIS ROUTINE IS CALLED TO SAVE THE DEVICE REGISTERS IN A SPECIFIED
03CF 764 : BUFFER. IT IS CALLED FROM THE DEVICE ERROR LOGGING ROUTINE.
03CF 765 :
03CF 766 : INPUTS:
03CF 767 : R0 = ADDRESS OF REGISTER SAVE BUFFER
03CF 768 : R4 = ADDRESS OF DEVICE CONTROL STATUS REGISTER
03CF 769 : R5= ADDRESS OF UNIT CONTROL BLOCK
03CF 770 : OUTPUTS:
03CF 771 : DEVICE REGISTERS ARE MOVED INTO THE SPECIFIED BUFFER.
03CF 772 : R0 CONTAINS THE ADDRESS OF THE NEXT EMPTY LONGWORD IN THE BUFFER.
03CF 773 : ALL REGISTERS EXCEPT R1 AND R2 ARE PRESERVED.
03CF 774 :
03CF 775 :--
03CF 776 LP_REGDUMP:
51 52 09 D0 03CF 777 MOVL #LP_NUM_REGS,R2 ; Store number of registers
80 52 D0 03D2 778 MOVL R2,(R0)+ ; Insert # of longwords dumped
00A0 C5 3E 03D5 779 MOVAW UCB$W_CSR(R5),R1 ; Get address of first reg.
80 81 3C 03DA 780 10$: MOVZWL (R1)+(R0)+ ; Dump registers into buffer
FA 52 F5 03DD 781 SOBGTR R2,10$ ; If GTR - more to go
05 03E0 782 RSB ; Return
03E1 783
```



```
03E1 785 .SBTTL FALLBACK - Macros to create character table for fallback
03E1 786
03E1 787 :
03E1 788 : Pointer to the fallback tables
03E1 789 :
03E1 790 TRANS TAB:
000003E5' 03E1 791 .LONG FALLTAB
03E5 792
03E5 793
03E5 794 :++
03E5 795 :FALLBACK - TABLE TO ALLOW THE TERMINAL TO DO FALLBACK PRESENTATION OF
03E5 796 : 8BIT CHARACTERS on 7 bit terminals
03E5 797 :
03E5 798 : Description:
03E5 799 : The following macros generate 1 table. The table is a 256 byte
03E5 800 : table with the single character fallback representation of all the
03E5 801 : characters that can be represented by a single character, those with
03E5 802 : no fallback presentation at all are represented by the _ character,
03E5 803 :
03E5 804 :--
03E5 805 .macro $fallini
03E5 806 $$=0
03E5 807 .repeat 256
03E5 808 .IF LE $$-<^X9F> ; EVERYTHING BUT THE MULTINATIONAL SET SHOULD
03E5 809 ; ECHO AS ITSELF.
03E5 810 .byte $$
03E5 811 .IFF
03E5 812 .BYTE ^A/_/
03E5 813 .ENDC
03E5 814 $$=$$+1
03E5 815 .endr
03E5 816 $$$=.
03E5 817 .endm $fallini
```

```

03E5 819 :++
03E5 820 : $FALL - generates the table entry for a given character
03E5 821 :
03E5 822 : Inputs:
03E5 823 :
03E5 824 :     CHARH - COLUMN IN THE ASCII TABLE.
03E5 825 :     CHARL - ROW IN THE ASCII TABLE.
03E5 826 :     FALLBACK - String that is the fallback representation
03E5 827 :     COUNT - Number of times to repeat this character
03E5 828 :--
03E5 829 :.MACRO $FALL CHARH,CHARL,FALLBACK,COUNT=1
03E5 830 :.=FALLTAB+<CHARH*16>+CHARL
03E5 831 :.REPEAT COUNT
03E5 832 :.NCHR SLEN,^FALLBACK\
03E5 833 :.IF EQ SLEN-1
03E5 834 :.BYTE ^A/FALLBACK/
03E5 835 :.ENDR
03E5 836 :.ENDM $FALL

```



```

03E5 838 :++
03E5 839 : $FALLEND - GENERATES END CONDITIONS FOR THE FALLBACK TABLE
03E5 840 :
03E5 841 : Description:
03E5 842 :
03E5 843 :     Resets the . to the end of the fallback table
03E5 844 :
03E5 845 : Inputs:
03E5 846 :
03E5 847 :     None
03E5 848 : --
03E5 849 : .MACRO $FALLEND
03E5 850 : .=$$$
03E5 851 : .ENDM $FALLEND

```

```

03E5 853
03E5 854 FALLTAB::
03E5 855 $FALLINI
04E5 856 $FALL 10.1.!
0487 857 $FALL 10.2.c
0488 858 $FALL 10.3.L
0489 859 $FALL 10.5.Y
048B 860 $FALL 10.8.O
048E 861 $FALL 10.10.a
0490 862 $FALL 11.0.o
0496 863 $FALL 11.1.+
0497 864 $FALL 11.2.2
0498 865 $FALL 11.3.3
0499 866 $FALL 11.5.u
049B 867 $FALL 11.7.;
049D 868 $FALL 11.9.i
049F 869 $FALL 11.10.o
04A0 870 $FALL 11.15.?
04A5 871 $FALL 12.0.A.6
04AB 872 $FALL 12.7.C
04AD 873 $FALL 12.8.E.4
04B1 874 $FALL 12.12.I.4
04B5 875 $FALL 13.1.N
04B7 876 $FALL 13.2.O.5
04BC 877 $FALL 13.8.O
04BE 878 $FALL 13.9.U.4
04C2 879 $FALL 13.13.Y
04C3 880 $FALL 14.0.a.6
04CB 881 $FALL 14.7.c
04CD 882 $FALL 14.8.e.4
04D1 883 $FALL 14.12.i.4
04D5 884 $FALL 15.1.n
04D7 885 $FALL 15.2.o.5
04DC 886 $FALL 15.8.o
04DE 887 $FALL 15.9.u.4
04E2 888 $FALL 15.13.y
04E3 889 $FALLEND
04E5 890
04E5 891
04E5 892 LP_END:
04E5 893
04E5 894 .END

```

; ADDRESS OF LAST LOCATION IN DRIVER



|   |          |    |    |
|---|----------|----|----|
| = | 0000003F |    |    |
| = | 00000020 |    |    |
| = | 0000000B |    |    |
| = | 00000030 |    |    |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
|   | *****    | X  | 03 |
| = | 0000003C |    |    |
| = | 00000038 |    |    |
| = | 0000002C |    |    |
| = | 00000001 |    |    |
| = | 00000006 |    |    |
| = | 00000000 |    |    |
| = | 00000000 |    |    |
| = | 00000030 |    |    |
| = | 00000020 |    |    |
| = | 0000002A |    |    |
| = | 00000020 |    |    |
|   | 000003AC | R  | 03 |
|   | 0000035A | R  | 03 |
| = | 00000005 |    |    |
| = | 00000042 |    |    |
|   | 00000000 | RG | 03 |
|   | 00000339 | RG | 03 |
| = | 00000080 |    |    |
| = | 00000002 |    |    |
| = | 00000009 |    |    |
| = | 00000008 |    |    |
|   | 00000000 |    |    |
| = | 00001000 |    |    |
| = | 00009000 |    |    |
| = | 00000040 |    |    |
| = | 00004000 |    |    |
| = | 00008000 |    |    |
| = | 00000001 |    |    |
| = | 0000000E |    |    |
|   | 00000002 |    |    |
|   | 000004E5 | R  | 03 |
| = | 00000076 |    |    |
| = | 00000780 |    |    |
| = | 00000009 |    |    |
|   | 000003CF | R  | 03 |
|   | 0000006C | R  | 03 |
| = | 00000080 |    |    |
| = | 08000000 |    |    |
|   | *****    | X  | 03 |
| = | 00000005 |    |    |
| = | 00000204 |    |    |
| = | 00000000 |    |    |
| = | 00000004 |    |    |

LCDRIVER  
Symbol table

- LINE PRINTER DRIVER FOR LP-730 H 7

16-SEP-1984 00:00:36 VAX/VMS Macro V04-00  
5-SEP-1984 00:14:49 [DRIVER.SRC]LCDRIVER.MAR;1

Page 27  
(5)

|                 |            |   |    |
|-----------------|------------|---|----|
| P3              | = 00000008 |   |    |
| P4              | = 0000000C |   |    |
| P5              | = 00000010 |   |    |
| P6              | = 00000014 |   |    |
| PAGSIZ          | = 00000200 |   |    |
| PCBSL_JIB       | = 00000080 |   |    |
| PCBSW_BIOCNT    | = 0000003A |   |    |
| PCBSW_DIOCNT    | = 0000003E |   |    |
| PR\$ IPL        | = 00000012 |   |    |
| RESET           | = 00000002 |   |    |
| SIZ...          | = 00000001 |   |    |
| SLEN            | = 00000001 |   |    |
| SS\$ ABORT      | = 0000002C |   |    |
| SS\$ BADPARAM   | = 00000014 |   |    |
| SS\$ NORMAL     | = 00000001 |   |    |
| STARTIO         | 0000011D   | R | 03 |
| SY\$GL OPRMBX   | *****      | X | 03 |
| TRANS_TAB       | 000003E1   | R | 03 |
| UCBSB_CSR2      | 000000AF   |   |    |
| UCBSB_DEVCLASS  | = 00000040 |   |    |
| UCBSB_DEVTYP    | = 00000041 |   |    |
| UCBSB_DIPL      | = 0000005E |   |    |
| UCBSB_FIPL      | = 0000000B |   |    |
| UCBSB_HIBUFADR  | 000000AE   |   |    |
| UCBSK_SIZE      | = 000000C4 |   |    |
| UCBSL_CRB       | = 00000024 |   |    |
| UCBSL_DEVCHAR   | = 00000038 |   |    |
| UCBSL_DEVCHAR2  | = 0000003C |   |    |
| UCBSL_DEVDEPEND | = 00000044 |   |    |
| UCBSL_DPC       | = 0000009C |   |    |
| UCBSL_FPC       | = 0000000C |   |    |
| UCBSL_FR3       | = 00000010 |   |    |
| UCBSL_IRP       | = 00000058 |   |    |
| UCBSL_MAPREG    | 000000C0   |   |    |
| UCBSL_REGSARE   | 000000B8   |   |    |
| UCBSL_SVAPTE    | = 00000078 |   |    |
| UCBSM_CANCEL    | = 00000008 |   |    |
| UCBSM_ONLINE    | = 00000010 |   |    |
| UCBSV_INT       | = 00000001 |   |    |
| UCBSV_POWER     | = 00000005 |   |    |
| UCBSW_BCNT      | = 0000007E |   |    |
| UCBSW_BOFF      | = 0000007C |   |    |
| UCBSW_BUFADR    | 000000AA   |   |    |
| UCBSW_BYTECNT   | 000000AC   |   |    |
| UCBSW_BYTXFR    | 000000A2   |   |    |
| UCBSW_CSR       | 000000A0   |   |    |
| UCBSW_DEVBUFSIZ | = 00000042 |   |    |
| UCBSW_LINCNT    | 000000A4   |   |    |
| UCBSW_LINES     | 000000B0   |   |    |
| UCBSW_LPTIMEOUT | 000000B2   |   |    |
| UCBSW_LP OFLCNT | 000000B4   |   |    |
| UCBSW_MAXBYT    | 000000B6   |   |    |
| UCBSW_PREFIX    | 000000A6   |   |    |
| UCBSW_STS       | = 00000064 |   |    |
| UCBSW_SUFFIX    | 000000A8   |   |    |
| VASS_BYTE       | = 00000009 |   |    |
| VASS_VPN        | = 00000015 |   |    |

|                |            |   |    |
|----------------|------------|---|----|
| VASV_BYTE      | = 00000000 |   |    |
| VASV_VPN       | = 00000009 |   |    |
| VECSL_IDB      | = 00000008 |   |    |
| VECSL_INITIAL  | = 0000000C |   |    |
| VECSL_UNITINIT | = 00000018 |   |    |
| VECSM_MAPLOCK  | = 00008000 |   |    |
| VECSV_MAPLOCK  | = 0000000F |   |    |
| VECSW_MAPREG   | = 00000010 |   |    |
| WRITE          | 00000178   | R | 03 |
| WRITEP         | 0000017F   | R | 03 |



+-----+  
! Psect synopsis !  
+-----+

| PSECT name         | Allocation        | PSECT No. | Attributes                                              |
|--------------------|-------------------|-----------|---------------------------------------------------------|
| ABS                | 00000000 ( 0.)    | 00 ( 0.)  | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| \$ABSS             | 000000C4 ( 196.)  | 01 ( 1.)  | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE       |
| \$\$\$105_PROLOGUE | 00000077 ( 119.)  | 02 ( 2.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE       |
| \$\$\$115_DRIVER   | 000004E5 ( 1253.) | 03 ( 3.)  | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC LONG       |

+-----+  
! Performance indicators !  
+-----+

| Phase                  | Page faults | CPU Time    | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization         | 32          | 00:00:00.06 | 00:00:00.57  |
| Command processing     | 141         | 00:00:00.36 | 00:00:03.30  |
| Pass 1                 | 598         | 00:00:19.04 | 00:01:08.48  |
| Symbol table sort      | 0           | 00:00:02.53 | 00:00:09.51  |
| Pass 2                 | 164         | 00:00:03.67 | 00:00:12.70  |
| Symbol table output    | 22          | 00:00:00.15 | 00:00:01.23  |
| Psect synopsis output  | 3           | 00:00:00.02 | 00:00:00.01  |
| Cross-reference output | 0           | 00:00:00.00 | 00:00:00.00  |
| Assembler run totals   | 962         | 00:00:25.84 | 00:01:35.81  |

The working set limit was 1950 pages.

158585 bytes (310 pages) of virtual memory were used to buffer the intermediate code.

There were 130 pages of symbol table space allocated to hold 2348 non-local and 35 local symbols.

894 source lines were read in Pass 1, producing 20 object records in Pass 2.

50 pages of virtual memory were used to define 47 macros.

+-----+  
! Macro library statistics !  
+-----+

| Macro library name                  | Macros defined |
|-------------------------------------|----------------|
| _\$255\$DUA28:[SYS.OBJ]LIB.MLB;1    | 30             |
| _\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 11             |
| TOTALS (all libraries)              | 41             |

2587 GETS were required to define 41 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:LCDRIVER/OBJ=OBJ\$:LCDRIVER MSRC\$:LCDRIVER/UPDATE=(ENH\$:LCDRIVER)+EXECML\$/LIB



0112 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

