```
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEEEEEEEEEEEEEE  RRRRRRRRRRRR     FFFFFFFFFFFFFFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEE              RRR         RRR  FFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEEEEEEEEEEE     RRRRRRRRRRRR     FFFFFFFFFFFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR   RRR        FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEE              RRR      RRR     FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
EEEEEEEEEEEEEEE  RRR         RRR  FFF
```

```
LL              CCCCCCCC  PPPPPPPP    RRRRRRRR      IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
LL              CCCCCCCC  PPPPPPPP    RRRRRRRR      IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
LL            CC          PP      PP  RR      RR      II      NN      NN      TT      EE          RR      RR
LL            CC          PP      PP  RR      RR      II      NN      NN      TT      EE          RR      RR
LL            CC          PP      PP  RR      RR      II      NNNN    NN      TT      EE          RR      RR
LL            CC          PP      PP  RR      RR      II      NNNN    NN      TT      EE          RR      RR
LL            CC          PPPPPPPP    RRRRRRRR        II      NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
LL            CC          PPPPPPPP    RRRRRRRR        II      NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
LL            CC          PP          RR  RR          II      NN    NNNN      TT      EE          RR  RR
LL            CC          PP          RR  RR          II      NN    NNNN      TT      EE          RR  RR
LL            CC          PP          RR    RR        II      NN      NN      TT      EE          RR    RR
LL            CC          PP          RR    RR        II      NN      NN      TT      EE          RR    RR
LLLLLLLLLL      CCCCCCCC  PP          RR      RR    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR
LLLLLLLLLL      CCCCCCCC  PP          RR      RR    IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR


LL              IIIIII      SSSSSSSS
LL              IIIIII    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
LLLLLLLLLL      IIIIII    SSSSSSSS
LLLLLLLLLL      IIIIII    SSSSSSSS
```

```
0001    C
0002    C Version:      'V04-000'
0003    C
0004    C*****************************************************************************
0005    C*                                                                           *
0006    C*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0007    C*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0008    C*  ALL RIGHTS RESERVED.                                                     *
0009    C*                                                                           *
0010    C*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0011    C*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0012    C*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0013    C*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0014    C*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0015    C*  TRANSFERRED.                                                             *
0016    C*                                                                           *
0017    C*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0018    C*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0019    C*  CORPORATION.                                                             *
0020    C*                                                                           *
0021    C*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0022    C*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0023    C*                                                                           *
0024    C*                                                                           *
0025    C*****************************************************************************
0026    C
0027
0028    c       Author Brian Porter                     Creation Date 20-AUG-1981
0029
0030    c++
0031    c       Functional description:
0032    c
0033    c       This module formats errorlog entries made by the lcdriver.  The format
0034    c       of the buffer is as follows.
0035    c
0036    c       +-----------------------------------+
0037    c       |            ucb$w_csr              |
0038    c       +-----------------------------------+
0039    c       |           ucb$w_bytxfer           |
0040    c       +-----------------------------------+
0041    c       |            ucb$w_lincnt           |
0042    c       +-----------------------------------+
0043    c       |            ucb$w_prefix           |
0044    c       +-----------------------------------+
0045    c       |            ucb$w_suffix           |
0046    c       +-----------------------------------+
0047    c       |            ucb$w_bufadr           |
0048    c       +-----------------------------------+
0049    c       |           ucb$w_bytecnt           |
0050    c       +-----------------------------------+
0051    c       |            ucb$w_csr2             |
0052    c       +-----------------------------------+
0053    c       |            ucb$w_lines            |
0054    c       +-----------------------------------+
0055    c       |         final map register        |
0056    c       +-----------------------------------+
0057    c       |        previous map register      |
```

```
0058    c       +-----------------------------------+
0059    c       !            vec$l_mapreg           !
0060    c       +-----------------------------------+
0061    c
0062    c       Modified by:
0063    c
0064    C       V03-002 SAR0223         Sharon A. Reynolds,     28-Mar-1984
0065    C               Changed the call to UCB$L_OWNUIC to ORB$L_OWNER.
0066    C
0067    C       V03-001 SAR0076         Sharon A. Reynolds,     20-Jun-1983
0068    C               Changed the carriage control in the 'format' statements
0069    C               for use with ERF.
0070    C
0071    c       v02-002 BP0002          Brian Porter,           23-NOV-1981
0072    c               Minor edit.
0073    c
0074    c       v02-001 BP0001          Brian Porter,           06-NOV-1981
0075    c               Added 'device attention' support.
0076    c**
0077    c--
0078
0079
0080
0081            subroutine lcprinter (lun)
0082
0083
0084
0085
0086            include 'src$:msghdr.for /nolist'
0145
0146            include 'src$:deverr.for /nolist'
0247
0248
0249
0250
0251            byte            lun
0252
0253            integer*4       csr
0254
0255            integer*4       bytxfer
0256
0257            integer*4       lincnt
0258
0259            integer*4       prefix
0260
0261            integer*4       suffix
0262
0263            integer*4       bufadr
0264
0265            integer*4       bytecnt
0266
0267            integer*4       csr2
0268
0269            integer*4       lines
0270
0271            integer*4       final_map_register
0272
```

```
0273            integer*4       previous_map_register
0274
0275            integer*4       vec$l_mapreg
0276
0277            equivalence     (emb$l_dv_regsav(0),csr)
0278
0279            equivalence     (emb$l_dv_regsav(1),bytxfer)
0280
0281            equivalence     (emb$l_dv_regsav(2),lincnt)
0282
0283            equivalence     (emb$l_dv_regsav(3),prefix)
0284
0285            equivalence     (emb$l_dv_regsav(4),suffix)
0286
0287            equivalence     (emb$l_dv_regsav(5),bufadr)
0288
0289            equivalence     (emb$l_dv_regsav(6),bytecnt)
0290
0291            equivalence     (emb$l_dv_regsav(7),csr2)
0292
0293            equivalence     (emb$l_dv_regsav(8),lines)
0294
0295            equivalence     (emb$l_dv_regsav(9),final_map_register)
0296
0297            equivalence     (emb$l_dv_regsav(10),previous_map_register)
0298
0299            equivalence     (emb$l_dv_regsav(11),vec$l_mapreg)
0300
0301            integer*4       compress4
0302
0303            integer*4       compressc
0304
0305            logical*1       diagnostic_mode
0306
0307            integer*4       indirect_address
0308
0309            integer*4       bytes_to_printer
0310
0311            integer*4       paper_lines_moved
0312
0313            byte            prefix_array(2)
0314
0315            equivalence     (prefix,prefix_array)
0316
0317            integer*4       prefix_value
0318
0319            integer*4       prefix_count
0320
0321            character*1     prefix_character
0322
0323            byte            suffix_array(2)
0324
0325            equivalence     (suffix,suffix_array)
0326
0327            integer*4       suffix_value
0328
0329            integer*4       suffix_count
```

```
0330
0331            character*1     suffix_character
0332
0333            integer*4       page_length
0334
0335            integer*4       carriage_width
0336
0337            integer*4       selected_map_register
0338
0339            integer*4       field
0340
0341            character*15    v1csr(0:2)
0342
0343            data            v1csr(0)        /'PRINT ENABLE*'/
0344
0345            data            v1csr(1)        /'MASTER RESET*'/
0346
0347            data            v1csr(2)        /'FORMAT CONTROL*'/
0348
0349            character*17    v2csr(6:7)
0350
0351            data            v2csr(6)        /'INTERRUPT ENABLE*'/
0352
0353            data            v2csr(7)        /'PRINT DONE*'/
0354
0355            character*27    v3csr(12:15)
0356
0357            data            v3csr(12)       /'LINE PRINTER NOT CONNECTED*'/
0358
0359            data            v3csr(13)       /'"DAVFU" READY*'/
0360
0361            data            v3csr(14)       /'LINE PRINTER OFF-LINE*'/
0362
0363            data            v3csr(15)       /'NON-EXISTENT MEMORY*'/
0364
0365            character*28    v1csr2(0:1)
0366
0367            data            v1csr2(0)       /'EXTENDED BUS ADDRESS BIT 16*'/
0368
0369            data            v1csr2(1)       /'EXTENDED BUS ADDRESS BIT 17*'/
0370
0371            character*31    v2csr2(8:10)
0372
0373            data            v2csr2(8)       /'AUTO INSERT <CR>*'/
0374
0375            data            v2csr2(9)       /'<FF> TO <LF> CONVERT*'/
0376
0377            data            v2csr2(10)      /'NON-PRINTING BYTE(S) DISCARDED*'/
0378
0379            character*10    v3csr2(11:12)
0380
0381            data            v3csr2(11)      /'"DAVFU"*'/
0382
0383            data            v3csr2(12)      /'LINE WRAP*'/
0384
0385            character*33    v4csr2(15:15)
0386
```

```
0387            data            v4csr2(15)      /'LOWER CASE TO UPPER CASE CONVERT*'/
0388
0389            character*8     reg_herald(0:8)
0390
0391            data            reg_herald(0)   /'CSR*'/
0392
0393            data            reg_herald(1)   /'BYTXFER*'/
0394
0395            data            reg_herald(2)   /'LINCNT*'/
0396
0397            data            reg_herald(3)   /'PREFIX*'/
0398
0399            data            reg_herald(4)   /'SUFFIX*'/
0400
0401            data            reg_herald(5)   /'BUFADR*'/
0402
0403            data            reg_herald(6)   /'BYTECNT*'/
0404
0405            data            reg_herald(7)   /'CSR2*'/
0406
0407            data            reg_herald(8)   /'LINES*'/
0408
0409
0410
0411            call frctof (lun)
0412
0413            call dhead1 (lun,'UBA DMF-32')
0414
0415            diagnostic_mode = .false.
0416
0417            if (lib$extzv (5,1,csr) .eq. 1) diagnostic_mode = .true.
0418
0419            call linchk (lun,2)
0420
0421            write(lun,10) 'CSR',csr
0422    10      format(/' ',t8,a,t24,z8.8)
0423
0424            if (.not. diagnostic_mode) then
0425
0426            call output (lun,csr,v1csr,0,0,2,'0')
0427
0428            call output (lun,csr,v2csr,6,6,7,'0')
0429
0430            indirect_add   s  = lib$extzv (8,3,csr)
0431
0432            call linchk (lun,1)
0433
0434            write(lun,15) 'INDIRECT ADDRESS #',indirect_address,'.'
0435    15      format(' ',t40,a,i<compress4 (indirect_address)>,a)
0436
0437            call output (lun,csr,v3csr,12,12,15,'0')
0438            else
0439
0440            call linchk (lun,1)
0441
0442            write(lun,20) 'DIAGNOSTIC MODE'
0443    20      format(' ',t40,a)
```

```
0444            endif
0445
0446            call linchk (lun,1)
0447
0448            write(lun,25) 'BYTXFER',bytxfer
0449    25      format(' ',t8,a,t24,z8.8)
0450
0451            if (.not. diagnostic_mode) then
0452
0453            bytes_to_printer = lib$extzv(0,16,bytxfer)
0454
0455            call linchk (lun,1)
0456
0457            write(lun,30) bytes_to_printer,'. BYTE(S) TO PRINTER'
0458    30      format(' ',t40,i<compress4 (bytes_to_printer)>,
0459            endif
0460
0461            call linchk (lun,1)
0462
0463            write(lun,25) 'LINCNT',lincnt
0464
0465            if (.not. diagnostic_mode) then
0466
0467            if (lib$extzv(11,1,csr2) .eq. 0) then
0468
0469            paper_lines_moved = lib$extzv(0,16,lincnt)
0470
0471            call linchk (lun,1)
0472
0473            write(lun,35) paper_lines_moved,'. PAPER LINE(S) THIS BUFFER'
0474    35      format(' ',t40,i<compress4 (paper_lines_moved)>,a)
0475            endif
0476            endif
0477
0478            call linchk (lun,1)
0479
0480            write(lun,25) 'PREFIX',prefix
0481
0482            if (.not. diagnostic_mode) then
0483
0484            prefix_count = lib$extzv(0,8,prefix)
0485
0486            prefix_value = lib$extzv(8,8,prefix)
0487
0488            prefix_character = char(prefix_array(2))
0489
0490            if (prefix_count .ne. 0) then
0491
0492            if (prefix_value .ne. 0) then
0493
0494            call linchk (lun,2)
0495
0496            write(lun,40) 'PREFIX COUNT ',prefix_count,'.'
0497    40      format(' ',t40,a,i<compress4 (prefix_count)>,a)
0498
0499            prefix_character = '.'
0500
```

```
0501            call sys$fao ('!af',,prefix_character,%val(1),prefix_value)
0502
0503            if (prefix_character .eq. '.') then
0504
0505            write(lun,20) 'PREFIX CHARACTER NON-PRINTABLE'
0506            else
0507
0508            write(lun,20) 'PREFIX CHARACTER '''//prefix_character//''''
0509            endif
0510
0511            else if (
0512            1 prefix_value .eq. 0
0513            1 .and.
0514            1 lib$extzv(8,1,csr2) .eq. 1
0515            1 ) then
0516
0517            call linchk (lun,1)
0518
0519            write(lun,45) prefix_count,'. <CR><LF> SEQUENCE(S) PREFIXED'
0520     45     format(' ',t40,i<compress4 (prefix_count)>,a)
0521
0522            else if (prefix_value .eq. 0) then
0523
0524            call linchk (lun,1)
0525
0526            write(lun,45) prefix_count,'. <LF> SEQUENCE(S) PREFIXED'
0527            endif
0528            endif
0529            endif
0530
0531            call linchk (lun,1)
0532
0533            write(lun,25) 'SUFFIX',suffix
0534
0535            if (.not. diagnostic_mode) then
0536
0537            suffix_count = lib$extzv(0,8,suffix)
0538
0539            suffix_character = char(suffix_array(2))
0540
0541            suffix_value = lib$extzv(8,8,suffix)
0542
0543            if (suffix_count .ne. 0) then
0544
0545            if (suffix_value .ne. 0) then
0546
0547            call linchk (lun,2)
0548
0549            write(lun,50) 'SUFFIX COUNT ',suffix_count,'.'
0550     50     format(' ',t40,a,i<compress4 (suffix_count)>,a)
0551
0552            suffix_character = '.'
0553
0554            call sys$fao ('!af',,suffix_character,%val(1),suffix_value)
0555
0556            if (suffix_character .eq. '.') then
0557
```

```
0558            write(lun,20) 'SUFFIX CHARACTER NON-P.INTABLE'
0559            else
0560
0561            write(lun,20) 'SUFFIX CHARACTER '''//suffix_character//''''
0562            endif
0563
0564            else if (
0565            1 suffix_value .eq. 0
0566            1 .and.
0567            1 lib$extzv(8,1,csr2) .eq. 1
0568            1 ) then
0569
0570            call linchk (lun,1)
0571
0572            write(lun,55) suffix_count,'. <CR><LF> SEQUENCE(S) SUFFIXED'
0573     55     format(' ',t40,i<compress4 (suffix_count)>,a)
0574
0575            else if (suffix_value .eq. 0) then
0576
0577            call linchk (lun,1)
0578
0579            write(lun,55) suffix_count,'. <LF> SEQUENCE(S) SUFFIXED'
0580            endif
0581            endif
0582            endif
0583
0584            call linchk (lun,1)
0585
0586            write(lun,25) 'BUFADR',bufadr
0587
0588            if (.not. diagnostic_mode) then
0589
0590            call calc_map (lun,0,csr2,bufadr)
0591            endif
0592
0593            call linchk (lun,1)
0594
0595            write(lun,25) 'BYTECNT',bytecnt
0596
0597            call linchk (lun,1)
0598
0599            write(lun,25) 'CSR2',csr2
0600
0601            if (.not. diagnostic_mode) then
0602
0603            call output (lun,csr2,v1csr2,0,0,1,'0')
0604
0605            if (lib$extzv(2,1,csr) .eq. 1) then
0606
0607            call output (lun,csr2,v2csr2,8,8,10,'1')
0608            endif
0609
0610            call output (lun,csr2,v3csr2,11,11,12,'0')
0611
0612            call output (lun,csr2,v4csr2,15,15,15,'0')
0613            endif
0614
```

```
0615            call linchk (lun,1)
0616
0617            write(lun,25) 'LINES',lines
0618
0619            if (.not. diagnostic_mode) then
0620
0621            if (
0622            1 lib$extzv(2,1,csr) .eq. 1
0623            1 .and.
0624            1 lib$extzv(11,1,csr2) .eq. 0
0625            1 ) then
0626
0627            page_length = lib$extzv(0,8,lines)
0628
0629            carriage_width = lib$extzv(8,8,lines)
0630
0631            call linchk (lun,2)
0632
0633            write(lun,60) 'PAGE LENGTH, ',page_length,'. LINE(S)',
0634            1 'PAGE WIDTH, ',carriage_width,'. CHARACTER(S)'
0635    60      format(' ',t40,a,i<compress4 (page_length)>,a,/,
0636            1 t40,a,i<compress4 (carriage_width)>,a)
0637            endif
0638            endif
0639
0640            if (emb$w_hd_entry .ne. 98) then
0641
0642    c       call calc_map2 (0,csr2,bufadr,selected_map_register)
0643
0644    c       call uba_mapping (lun,selected_map_register,final_map_register)
0645
0646    c       if (lib$extzv(16,16,emb$l_dv_iosb1) .gt. 512) then
0647
0648    c       call uba_mapping (lun,(selected_map_register-1),previous_map_register)
0649    c       endif
0650
0651    c       call vecmapreg (lun,vec$l_mapreg)
0652            endif
0653
0654            call linchk (lun,1)
0655
0656            write(lun,65)
0657    65      format(' ',:)
0658
0659            call orb$l_owner (lun,emb$l_dv_ownuic)
0660
0661            call ucb$l_char (lun,emb$l_dv_char)
0662
0663            call ucb$w_sts (lun,emb$w_dv_sts)
0664
0665            call ucb$l_opcnt (lun,emb$l_dv_opcnt)
0666
0667            call ucb$w_errcnt (lun,emb$w_dv_errcnt)
0668
0669            if (emb$w_hd_entry .ne. 98) then
0670
0671            call linchk (lun,1)
```

```
0672
0673            write(lun,65)
0674
0675            call lcprinter_qio (lun,emb$w_dv_func)
0676
0677            call irp$w_bcnt (lun,emb$w_dv_bcnt)
0678
0679            call irp$w_boff (lun,emb$w_dv_boff)
0680
0681            call irp$l_pid (lun,emb$l_dv_rqpid)
0682
0683            call irp$q_iosb (lun,emb$l_dv_iosb1)
0684            endif
0685
0686            return
0687
0688
0689
0690            entry b_lcprinter (lun)
0691
0692
0693
0694            call dhead1 (lun,'UBA DMF-32')
0695
0696            call brief32 (lun,(9),csr,reg_herald,emb$t_dv_name,emb$w_dv_unit)
0697
0698            return
0699
0700
0701
0702            entry c_lcprinter (lun)
0703
0704
0705
0706            call cryptk (lun,32,(9),csr,reg_herald,emb$t_dv_name,emb$w_dv_unit)
0707
0708            return
0709
0710            end
```

PPOGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	2467	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	648	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	1832	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	5459	

ENTRY POINTS

Address	Type	Name	Address	Type	Name	Address	Type	Name
0-000008D6		B_LCPRINTER	0-00000905		C_LCPRINTER	0-00000000		LCPRINTER

VARIABLES

Address	Type	Name	Address	Type	Name
3-00000066	I*4	BUFADR	3-0000006A	I*4	BYTECNT
2-000001D8	I*4	BYTES_TO_PRINTER	3-00000056	I*4	BYTXFER
2-000001F4	I*4	CARRIAGE_WIDTH	2-000001D0	I*4	COMPRESSC
3-00000052	I*4	CSR	3-0000006E	I*4	CSR2
2-000001CD	L*1	DIAGNOSTIC_MODE	3-0000001C	L*1	EMB$B_DV_CLASS
3-00000010	L*1	EMB$B_DV_ERTCNT	3-00000011	L*1	EMB$B_DV_ERTMAX
3-0000003E	L*1	EMB$B_DV_NAMLNG	3-0000003A	L*1	EMB$B_DV_SLAVE
3-0000001D	L*1	EMB$B_DV_TYPE	3-00000036	I*4	EMB$L_DV_CHAR
3-00000012	I*4	EMB$L_DV_IOSB1	3-00000016	I*4	EMB$L_DV_IOSB2
3-00000026	I*4	EMB$L_DV_MEDIA	3-0000004E	I*4	EMB$L_DV_NUMREG
3-0000002E	I*4	EMB$L_DV_OPCNT	3-00000032	I*4	EMB$L_DV_OWNUIC
3-0000001E	I*4	EMB$L_DV_RQPID	3-00000000	I*4	EMB$L_HD_SID
3-0000003F	CHAR	EMB$T_DV_NAME	3-00000024	I*2	EMB$W_DV_BCNT
3-00000022	I*2	EMB$W_DV_BOFF	3-0000002C	I*2	EMB$W_DV_ERRCNT
3-0000003C	I*2	EMB$W_DV_FUNC	3-0000001A	I*2	EMB$W_DV_STS
3-0000002A	I*2	EMB$W_DV_UNIT	3-00000004	I*2	EMB$W_HD_ENTRY
3-0000000E	I*2	EMB$W_HD_ERRSEQ	2-000001FC	I*4	FIELD
3-00000076	I*4	FINAL_MAP_REGISTER	2-000001D4	I*4	INDIRECT_ADDRESS
3-0000005A	I*4	LINCNT	3-00000072	I*4	LINES
AP-0000004@	L*1	LUN	2-000001F0	I*4	PAGE_LENGTH
2-000001DC	I*4	PAPER_LINES_MOVED	3-0000005E	I*4	PREFIX
2-000001CE	CHAR	PREFIX_CHARACTER	2-000001E4	I*4	PREFIX_COUNT
2-000001E0	I*4	PREFIX_VALUE	3-0000007A	I*4	PREVIOUS_MAP_REGISTER
2-000001F8	I*4	SELECTED_MAP_REGISTER	3-00000062	I*4	SUFFIX
2-000001CF	CHAR	SUFFIX_CHARACTER	2-000001EC	I*4	SUFFIX_COUNT
2-000001E8	I*4	SUFFIX_VALUE	3-0000007E	I*4	VEC$L_MAPREG

ARRAYS

Address	Type	Name	Bytes	Dimensions
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMB$L_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMB$Q_HD_TIME	8	(2)
3-0000005E	L*1	PREFIX_ARRAY	2	(2)
2-00000185	CHAR	REG_HERALD	72	(0:8)
3-00000062	L*1	SUFFIX_ARRAY	2	(2)
2-00000000	CHAR	V1CSR	45	(0:2)
2-000000BB	CHAR	V1CSR2	56	(0:1)
2-0000002D	CHAR	V2CSR	34	(6:7)
2-000000F3	CHAR	V2CSR2	93	(8:10)
2-0000004F	CHAR	V3CSR	108	(12:15)
2-00000150	CHAR	V3CSR2	20	(11:12)
2-00000164	CHAR	V4CSR2	33	(15:15)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-000001C8	10'	1-000001D5	15'	1-000001E3	20'	1-000001EA	25'	1-000001F6	30'	1-00000203	35'
1-00000210	40'	1-0000021E	45'	1-0000022B	50'	1-00000239	55'	1-00000246	60'	1-0000025F	65'

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
	BRIEF32		CALC_MAP	I*4	COMPRESS4		CRYPTK		DHEAD1		FRCTOF
	IRP$L_PID		IRP$Q_IOSB		IRP$W_BCNT		IRP$W_BOFF		LCPRINTER_QIO	I*4	LIB$EXTZV
	LINCHR		ORB$L_OWNER		OUTPUT		SYS$FAO		UCB$L_CHAR		UCB$L_OPCNT
	UCB$W_ERRCNT		UCB$W_STS								

```
0001
0002
0003
0004
0005          subroutine lcprinter_qio (lun,emb$w_dv_func)
0006
0007
0008
0009          include 'src$:qiocommon.for /nolist'
0273
0274
0275
0276          byte            lun
0277
0278          integer*2       emb$w_dv_func
0279
0280          integer*4       qiocode(0:1,0:63)
0281
0282
0283
0284
0285          if (qiocode(0,0) .eq. 0) then
0286
0287          qiocode(1,11) = %loc(io$_writepblk)
0288
0289          qiocode(1,26) = %loc(io$_setchar)
0290
0291          qiocode(1,27) = %loc(io$_sensechar)
0292
0293          qiocode(1,32) = %loc(io$_writelblk)
0294
0295          qiocode(1,35) = %loc(io$_setmode)
0296
0297          qiocode(1,39) = %loc(io$_sensemode)
0298
0299          qiocode(1,48) = %loc(io$_writevblk)
0300
0301          do 10,i = 0,63
0302
0303          qiocode(0,i) = 33
0304
0305          if (qiocode(1,i) .eq. 0) then
0306
0307          qiocode(1,i) = %loc(qio_string)
0308          endif
0309
0310    10    continue
0311          endif
0312
0313          call irp$w_func (lun,emb$w_dv_func,
0314          1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0315
0316          return
0317
0318          end
```

PROGRAM SECTIONS

	Name	Bytes	Attributes
0	$CODE	146	PIC CON REL LCL SHR EXE RD NOWRT LONG
1	$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2	$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3	QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
	Total Space Allocated	1949	

ENTRY POINTS

Address	Type	Name
0-00000000		LCPRINTER_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMBSW_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IO$_ABORT	3-0000034D	CHAR	IO$_ACCESS
3-000003C2	CHAR	IO$_ACPCONTROL	3-000004B3	CHAR	IO$_AVAILABLE
3-00000297	CHAR	IO$_CLEAN	3-00000369	CHAR	IO$_CREATE
3-00000385	CHAR	IO$_DEACCESS	3-00000393	CHAR	IO$_DELETE
3-0000026D	CHAR	IO$_DIAGNOSE	3-00000065	CHAR	IO$_DRVCLR
3-000004CB	CHAR	IO$_DSE	3-000000A9	CHAR	IO$_ERASETAPE
3-00000276	CHAR	IO$_FORMAT	3-00000071	CHAR	IO$_INITIALIZE
3-00000014	CHAR	IO$_LOADMCODE	3-000003A1	CHAR	IO$_MODIFY
3-000003E2	CHAR	IO$_MOUNT	3-00000000	CHAR	IO$_NOP
3-0000009D	CHAR	IO$_OFFSET	3-000000EB	CHAR	IO$_PACKACK
3-000000E0	CHAR	IO$_QSTOP	3-000003EF	CHAR	IO$_RDSTATS
3-00000421	CHAR	IO$_READCSR	3-00000169	CHAR	IO$_READHEAD
3-00000286	CHAR	IO$_READLBLK	3-0000013F	CHAR	IO$_READPBLK
3-00000200	CHAR	IO$_READPRESET	3-00000195	CHAR	IO$_READTRACKD
3-0000033A	CHAR	IO$_READVBLK	3-0000045A	CHAR	IO$_READWTHBUF
3-00000484	CHAR	IO$_READWTHXBUF	3-0000004D	CHAR	IO$_RECAL
3-0000007C	CHAR	IO$_RELEASE	3-000001AB	CHAR	IO$_REREADN
3-000001B8	CHAR	IO$_REREADP	3-000000CA	CHAR	IO$_RETCENTER
3-000002E6	CHAR	IO$_REWIND	3-000002C9	CHAR	IO$_REWINDOFF
3-000000FC	CHAR	IO$_SEARCH	3-00000024	CHAR	IO$_SEEK
3-00000231	CHAR	IO$_SENSECHAR	3-00000309	CHAR	IO$_SENSEMODE
3-0000021D	CHAR	IO$_SETCHAR	3-000003B8	CHAR	IO$_SETCLOCK
3-00000088	CHAR	IO$_SETCLOCKP	3-000002DD	CHAR	IO$_SETMODE
3-000002ED	CHAR	IO$_SKIPFILE	3-000002FA	CHAR	IO$_SKIPRECORD
3-00000029	CHAR	IO$_SPACEFILE	3-0000010E	CHAR	IO$_SPACERECORD
3-000003D7	CHAR	IO$_STARTDATA	3-000000B4	CHAR	IO$_STARTDATAP
3-00000037	CHAR	IO$_STARTMPROC	3-0000020F	CHAR	IO$_STARTSPNDL
3-00000059	CHAR	IO$_STOP	3-0000000D	CHAR	IO$_UNLOAD
3-00000468	CHAR	IO$_WRITEBUFNCRC	3-0000011E	CHAR	IO$_WRITECHECK
3-000001E4	CHAR	IO$_WRITECHECKH	3-000003FF	CHAR	IO$_WRITECSR
3-00000153	CHAR	IO$_WRITEHEAD	3-000002A2	CHAR	IO$_WRITELBLK
3-00000247	CHAR	IO$_WRITEMARK	3-00000314	CHAR	IO$_WRITEOF
3-0000012A	CHAR	IO$_WRITEPBLK	3-000001C9	CHAR	IO$_WRITERET

```
  3-0000017E  CHAR IO$_WRITETRACKD          3-00000326  CHAR IO$_WRITEVBLK
  3-00000448  CHAR IO$_WRITEWTHBUF          3-00000257  CHAR IO$_WRTTMKR
 AP-00000004a L*1  LUN                      3-000004A1  CHAR QIO_STRING

ARRAYS

    Address  Type  Name                Bytes  Dimensions

  2-00000000  I*4  QIOCODE               512  (0:1, 0:63)

LABELS

    Address   Label

      **      10

FUNCTIONS AND SUBROUTINES REFERENCED

  Type  Name              Type  Name

        IRP$W_FUNC         I*4  LIB$EXTZV

COMMAND QUALIFIERS

  FORTRAN /LIS=LIS$:LCPRINTER/OBJ=OBJ$:LCPRINTER MSRC$:LCPRINTER

  /CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)
  /DEBUG=(NOSYMBOLS,TRACEBACK)
  /STANDARD=(NOSYNTAX,NOSOURCE_FORM)
  /SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)
  /F77  /NOG_FLOATING  /I4  /OPTIMIZE  /WARNINGS  /NOD_LINES  /NOCROSS_REFERENCE  /NOMACHINE_CODE  /CONTINUATIONS=19


COMPILATION STATISTICS

  Run Time:              8.87 seconds
  Elapsed Time:         21.40 seconds
  Page Faults:          249
  Dynamic Memory:       239 pages
```

0150 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LPA11K
LIS

MCHECKS
LIS

LCPRINTER
LIS

MBA
LIS

LOGMSCP
LIS

LINCHK
LIS

MBAINT
LIS

LABEL
LIS

LOGGER
LIS

MASSTAPE
LIS

MASSDISK
LIS