```
FFFFFFFFFFFFFFFF        111               111           AAAAAAAAA
FFFFFFFFFFFFFFFF        111               111           AAAAAAAAA
FFFFFFFFFFFFFFFF        111               111           AAAAAAAAA
FFF                  111111           111111         AAA         AAA
FFF                  111111           111111         AAA         AAA
FFF                  111111           111111         AAA         AAA
FFF                     111               111        AAA         AAA
FFF                     111               111        AAA         AAA
FFF                     111               111        AAA         AAA
FFFFFFFFFFFF            111               111        AAA         AAA
FFFFFFFFFFFF            111               111        AAA         AAA
FFFFFFFFFFFF            111               111        AAA         AAA
FFF                     111               111        AAAAAAAAAAAAAAA
FFF                     111               111        AAAAAAAAAAAAAAA
FFF                     111               111        AAAAAAAAAAAAAAA
FFF                     111               111        AAA         AAA
FFF                     111               111        AAA         AAA
FFF                     111               111        AAA         AAA
FFF                  111111111        111111111      AAA         AAA
FFF                  111111111        111111111      AAA         AAA
FFF                  111111111        111111111      AAA         AAA
```

```
**FILE**ID**CREHDR                                          I 14
```

```
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
  CCCCCCCC  RRRRRRRR    EEEEEEEEEE  HH      HH  DDDDDDDD    RRRRRRRR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RR      RR  EE          HH      HH  DD      DD  RR      RR
CC          RRRRRRRR    EEEEEEEE    HHHHHHHHHH  DD      DD  RRRRRRRR
CC          RRRRRRRR    EEEEEEEE    HHHHHHHHHH  DD      DD  RRRRRRRR
CC          RR  RR      EE          HH      HH  DD      DD  RR  RR
CC          RR  RR      EE          HH      HH  DD      DD  RR  RR
CC          RR    RR    EE          HH      HH  DD      DD  RR    RR        ....
CC          RR    RR    EE          HH      HH  DD      DD  RR    RR        ....
  CCCCCCCC  RR      RR  EEEEEEEEEE  HH      HH  DDDDDDDD    RR      RR      ....
  CCCCCCCC  RR      RR  EEEEEEEEEE  HH      HH  DDDDDDDD    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
```

CR
VO

```
0 MODULE CREHDR (
0                 LANGUAGE (BLISS32),
0                 IDENT = 'V04-000'
0                 ) =
1 BEGIN
1
1 !
1 !****************************************************************************
1 !*                                                                          *
1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
1 !*  ALL RIGHTS RESERVED.                                                    *
1 !*                                                                          *
1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
1 !*  TRANSFERRED.                                                            *
1 !*                                                                          *
1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
1 !*  CORPORATION.                                                            *
1 !*                                                                          *
1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
1 !*                                                                          *
1 !*                                                                          *
1 !****************************************************************************
1
1 !++
1 !
1 ! FACILITY:  F11ACP Structure Level 1
1 !
1 ! ABSTRACT:
1 !
1 !       This routine creates a new file ID by allocating a file number from the
1 !       index file bitmap. It returns an empty file header, verified for use.
1 !
1 ! ENVIRONMENT:
1 !
1 !       STARLET operating system, including privileged system services
1 !       and internal exec routines.
1 !
1 !--
1 !
1 !
1 ! AUTHOR:  Andrew C. Goldstein,  CREATION DATE:  28-Mar-1977  13:49
1 !
1 ! MODIFIED BY:
1 !
1 !       A0101   ACG0117         Andrew C. Goldstein,    16-Jan-1980  17:07
1 !               Return true I/O status on I/O errors
1 !
1 !       A0100   ACG00001        Andrew C. Goldstein,  10-Oct-1978  20:02
1 !       Previous revision history moved to F11A.REV
1 !**
```

```
:    58          0058  1
:    59          0059  1
:    60          0060  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
:    61          0061  1 REQUIRE 'SRC$:FCPDEF.B32';
:    62          0376  1
:    63          0377  1
:    64          0378  1 FORWARD ROUTINE
:    65          0379  1         CREATE_HEADER,                  ! create file ID and header
:    66          0380  1         UPDATE_IBVBN    : NOVALUE,      ! update index bitmap scan start
:    67          0381  1         READ_NEW_HEADER,                ! read new file header block
:    68          0382  1         HANDLER;                        ! local condition handler
```

```
1 GLOBAL ROUTINE CREATE_HEADER =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine creates a new file ID by searching the volume's index
1 !       file bitmap for the first free file number. It also checks that a header
1 !       for the file number is present in the index file. It reads the old
1 !       header and establishes the file sequence number for the new one.
1 !
1 ! CALLING SEQUENCE:
1 !       CREATE_HEADER ()
1 !
1 ! INPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT INPUTS:
1 !       CURRENT_VCB: address of volume's VCB
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       address of buffer containing new header
1 !
1 ! SIDE EFFECTS:
1 !       VCB and index file bitmap altered, header block read
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       VCB             : REF BBLOCK,   ! local copy of VCB address
2       VBN,                            ! relative block number in bitmap
2       BUFFER          : REF BITVECTOR, ! address of index file bitmap buffer
2       ADDRESS         : REF BITVECTOR, ! address of byte in buffer
2       BITPOS,                         ! bit positon of free bit within byte
2       FILE_NUMBER,                    ! file number allocated
2       IDX_FCB         : REF BBLOCK,   ! FCB of index file
2       LBN,                            ! LBN of new file header
2       HEADER          : REF BBLOCK,   ! address of header buffer
2       SAVE_STATUS;                    ! save I/O status during CHECK_HEADER call
2
2 EXTERNAL
2       CURRENT_VCB     : REF BBLOCK,   ! VCB of volume
2       NEW_FID,                        ! pending file ID
2       HEADER_LBN,                     ! LBN of created file header
2       USER_STATUS     : VECTOR;       ! I/O status block of user
2
2 EXTERNAL ROUTINE
2       READ_BLOCK,                     ! read block from disk
2       WRITE_BLOCK,                    ! write block to disk
```

```
2         EXTEND_INDEX,                    ! extend the index file
2         MAP_VBN,                         ! map virtual to logical block
2         CHECK_HEADER;                    ! verify file header
2
2
2 ! The outer loop performs retries if blocks in the index file are bad.
2 ! Bad header blocks are simply left marked in use in the index file bitmap;
2 ! they will show up in a verify but are otherwise harmless.
2 !
2
2 VCB = .CURRENT_VCB;
2 WHILE 1 DO
3     BEGIN
3
3 ! We scan the index file bitmap for the first free (zero) bit. This is done
3 ! by starting with the block recorded in the VCB and looking at each block
3 ! with a character scan.
3 !
3
3     VBN = .VCB[VCB$B_IBMAPVBN];
3
3     IF
4         BEGIN
4         UNTIL .VBN GEQ .VCB[VCB$B_IBMAPSIZE] DO
5             BEGIN
5             BUFFER = READ_BLOCK (.VBN + .VCB[VCB$L_IBMAPLBN], 1, INDEX_TYPE);
5             IF NOT CH$FAIL (ADDRESS = CH$FIND_NOT_CH (512, .BUFFER, 255))
5             THEN EXITLOOP 0;
5             VBN = .VBN + 1;
5             END
4         END
4
4 ! If we fall through the loop, the entire bitmap is full.
4 !
4
3         THEN ERR_EXIT (SS$_IDXFILEFULL);
3
3 ! Having found a byte containing a zero bit, scan for the bit.
3 !
3
3     FFC (%REF (0), %REF (8), .ADDRESS, BITPOS);
3
3 ! Compute the file number and check it against the maximum files allowed
3 ! on the volume. Also check if the corresponding file header is present in
3 ! the index file. If not, extend the index file and re-read the bitmap
3 ! block, which may have been kicked out in the process.
3 !
3
3     FILE_NUMBER = .VBN*4096 + (.ADDRESS-.BUFFER)*8 + .BITPOS + 1;
3
3     IF .FILE_NUMBER GTR .VCB[VCB$L_MAXFILES]
3     THEN ERR_EXIT (SS$_IDXFILEFULL);
3
3     IDX_FCB = .VCB[VCB$L_FCBFL];
3     IF .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + 2 GTR .IDX_FCB[FCB$L_FILESIZE]
3     THEN
4         BEGIN
```

```
4           ADDRESS = .ADDRESS - .BUFFER;
4           EXTEND_INDEX (.FILE_NUMBER);
4           BUFFER = READ_BLOCK (.VBN + .VCB[VCB$L_IBMAPLBN], 1, INDEX_TYPE);
4           ADDRESS = .ADDRESS + .BUFFER;
3           END;
3
3 ! All is in order. Set the bit and rewrite the block. Also update the
3 ! scan point in the VCB for the next create. Note that if the file number was
3 ! from the last bit in the block, we will start the scan at the next block to
3 ! avoid wasting a read.
3 !
3
3       ADDRESS[.BITPOS] = 1;
3       WRITE_BLOCK (.BUFFER);
3
3       IF .FILE_NUMBER<0,12> EQL 0
3       THEN VBN = .VBN + 1;
3       KERNEL_CALL (UPDATE_IBVBN, .VBN);
3       NEW_FID = .FILE_NUMBER;                 ! record file ID for cleanup
3
3 ! Now read the old file header. If the block contained an old file header,
3 ! bump the file sequence number; else assign 1.
3 !
3
3       VBN = .FILE_NUMBER + .VCB[VCB$B_IBMAPSIZE] + 2;
3       IDX_FCB = .VCB[VCB$L_FCBFL];
3       LBN = MAP_VBN (.VBN, .IDX_FCB[FCB$L_WLFL]);
3       IF .LBN EQL -1 THEN BUG_CHECK (HDRNOTMAP, FATAL, 'Allocated file header not mapped');
3       HEADER = READ_NEW_HEADER (.LBN);
3       IF .HEADER NEQ 0 THEN EXITLOOP;
2       END;                                    ! end of file number allocation loop
2
2 HEADER_LBN = .LBN;                            ! record LBN of new header
2
2 SAVE_STATUS = .USER_STATUS[0];
2 IF CHECK_HEADER (.HEADER, UPLIT WORD (0, 0, 0)) NEQ 0
2 THEN HEADER[FH1$W_FID_SEQ] = .HEADER[FH1$W_FID_SEQ] + 1
2 ELSE HEADER[FH1$W_FID_SEQ] = 1;
2 HEADER[FH1$W_FID_NUM] = .FILE_NUMBER;
2 USER_STATUS[0] = .SAVE_STATUS;                ! restore status, bashed by CHECK_HEADER
2
2 RETURN .HEADER;
2
1 END;                                          ! end of routine CREATE_HEADER

                                                                    .TITLE  CREHDR
                                                                    .IDENT  \V04-000\

                                                                    .PSECT  $CODE$,NOWRT,2

                                   0000  0000  0000  00000 P.AAA:   .WORD   0, 0, 0                                  ;

                                                                    .EXTRN  CURRENT_VCB, NEW_FID
                                                                    .EXTRN  HEADER_LBN, USER_STATUS
                                                                    .EXTRN  READ_BLOCK, WRITE_BLOCK
                                                                    .EXTRN  EXTEND_INDEX, MAP_VBN
```

```
                                                                .EXTRN   CHECK_HEADER, SYS$CMKRNL
                                                                .EXTRN   BUG$_ADRNOTMAP

                                               0FFC 00000      .ENTRY   CREATE_HEADER, Save R2,R3,R4,R5,R6,R7,R8,-    ; 0383
                                                                         R9,R10,R11                                   ;
                              53     0000G  CF   D0 00002      MOVL     CURRENT_VCB, VCB                             ; 0450
                              52        3A  A3   9A 00007 1$:  MOVZBL   58(VCB), VBN                                 ; 0459
  52        38   A3           08            00   ED 0000B 2$:  CMPZV    #0, #8, 56(VCB), VBN                         ; 0463
                                            40   15 00011      BLEQ     5$                                           ;
                                            03   DD 00013      PUSHL    #3                                           ; 0465
                                            01   DD 00015      PUSHL    #1                                           ;
                                     30 B342     9F 00017      PUSHAB   @48(VCB)[VBN]                                ;
                      0000G   CF            03   FB 0001B      CALLS    #3, READ_BLOCK                               ;
                              5A            50   D0 00020      MOVL     R0, BUFFER                                   ;
               6A     0200    8F        FF  8F   3B 00023      SKPC     #255, #512, (BUFFER)                         ; 0466
                                            02   12 0002A      BNEQ     3$                                           ;
                                            51   D4 0002C      CLRL     R1                                           ;
                              59            51   D0 0002E 3$:  MOVL     R1, ADDRESS                                  ;
                                            04   12 00031      BNEQ     4$                                           ;
                                            52   D6 00033      INCL     VBN                                          ; 0468
                                            D4   11 00035      BRB      2$                                           ; 0463
  58                 69       08            00   EB 00037 4$:  FFC      #0, #8, (ADDRESS), BITPOS                    ; 0480
                     50       52            0C   78 0003C      ASHL     #12, VBN, R0                                 ; 0488
                     51       59            5A   C3 00040      SUBL3    BUFFER, ADDRESS, R1                          ;
                              50          6041   7E 00044      MOVAQ    (R0)[R1], R0                                 ;
                              55       01 A840   9E 00048      MOVAB    1(BITPOS)[R0], FILE_NUMBER                   ;
                        44    A3            55   D1 0004D      CMPL     FILE_NUMBER, 68(VCB)                         ; C490
                                            05   15 00051      BLEQ     6$                                           ;
                                     08D0   8F   BF 00053 5$:  CHMU     #2256                                        ; C491
                                                 04 00057      RET                                                   ;
                              56            63   D0 00058 6$:  MOVL     (VCB), IDX_FCB                               ; C493
                              50        38  A3   9A 0005B      MOVZBL   56(VCB), R0                                  ; 0494
                              57       02 A045   9E 0005F      MOVAB    2(R0)[FILE_NUMBER], R7                       ;
                        38    A6            57   D1 00064      CMPL     R7  56(IDX_FCB)                              ;
                                            1D   15 00068      BLEQ     7$                                           ;
                              59            5A   C2 0006A      SUBL?    BUFFER, ADDRESS                              ; 0497
                                            55   DD 0006D      PUSHL    FILE_NUMBER                                  ; 0498
                      0000G   CF            01   FB 0006F      CALLS    #1, EXTEND_INDEX                             ;
                                            03   DD 00074      PUSHL    #3                                           ; 0499
                                            01   DD 00076      PUSHL    #1                                           ;
                                     30 B342     9F 00078      PUSHAB   @48(VCB)[VBN]                                ;
                      0000G   CF            03   FB 0007C      CALLS    #3, READ_BLOCK                               ;
                              5A            50   D0 00081      MOVL     R0, BUFFER                                   ;
                              59            5A   C0 00084      ADDL2    BUFFER, ADDRESS                              ; 0500
               00             69            58   E2 00087 7$:  BBSS     BITPOS, (ADDRESS), 8$                        ; 0509
                                            5A   DD 0008B 8$:  PUSHL    BUFFER                                       ; 0510
                      0000G   CF            01   FB 0008D      CALLS    #1, WRITE_BLOCK                              ;
                      0FFF    8F            55   B3 00092      BITW     FILE_NUMBER, #4095                           ; 0512
                                            02   12 00097      BNEQ     9$                                           ;
                                            52   D6 00099      INCL     VBN                                          ; 0513
                                            52   DD 0009B 9$:  PUSHL    VBN                                          ; 0514
                                            01   DD 0009D      PUSHL    #1                                           ;
                                            5E   DD 0009F      PUSHL    SP                                           ;
                                     0000V  CF   9F 000A1      PUSHAB   UPDATE_IBVBN                                 ;
                  00000000G   9F            04   FB 000A5      CALLS    #4, @#SYS$CMKRNL                             ;
                      0000G   CF            55   D0 000AC      MOVL     FILE_NUMBER, NEW_FID                         ; 0515
                              52            57   D0 000B1      MOVL     R7, VBN                                      ; 0521
                        56                63 D0 000B4           MOVL    (VCB), IDX_FCB                                ; 0522
                                  10      A6 DD 000B7           PUSHL   16(IDX_FCB)                                   ; 0523
                                          52 DD 000BA           PUSHL   VBN                                           ;
                   0000G  CF              02 FB 000BC           CALLS   #2, MAP_VBN                                   ;
                          5B              50 D0 000C1           MOVL    R0, LBN                                       ;
               FFFFFFFF   8F              5B D1 000C4           CMPL    LBN, #-1                                      ; 0524
                                          04 12 000CB           BNEQ    10$                                           ;
                                           FEFF 000CD           BUGW                                                  ;
                                          0000* 000CF           .WORD   <BUG$_HDRNOTMAP!4>                            ;
                                          5B DD 000D1 10$:      PUSHL   LBN                                           ; 0525
                   0000V  CF              01 FB 000D3           CALLS   #1, READ_NEW_HEADER                           ;
                          54              50 D0 000D8           MOVL    R0, HEADER                                    ;
                                          03 12 000DB           BNEQ    11$                                           ; 0526
                                        FF27 31 000DD           BRW     1$                                            ;
                   0000G  CF              5B D0 000E0 11$:      MOVL    LBN, HEADER_LBN                               ; 0529
                          52     0000G    CF D0 000E5           MOVL    USER_STATUS, SAVE_STATUS                      ; 0531
                                 FF0C     CF 9F 000EA           PUSHAB  P.AAA                                         ; 0532
                                          54 DD 000EE           PUSHL   HEADER                                        ;
                   0000G  CF              02 FB 000F0           CALLS   #2, CHECK_HEADER                              ;
                                          50 D5 000F5           TSTL    R0                                            ;
                                          05 13 000F7           BEQL    12$                                           ;
                                  04      A4 B6 000F9           INCW    4(HEADER)                                     ; 0533
                                          04 11 000FC           BRB     13$                                           ;
                      04  A4              01 B0 000FE 12$:      MOVW    #1, 4(HEADER)                                 ; 0534
                      02  A4              55 B0 00102 13$:      MOVW    FILE_NUMBER, 2(HEADER)                        ; 0535
                   0000G  CF              52 D0 00106           MOVL    SAVE_STATUS, USER_STATUS                      ; 0536
                          50              54 D0 0010B           MOVL    HEADER, R0                                    ; 0538
                                             04 0010E           RET                                                   ; 0540

; Routine Size:  271 bytes,    Routine Base:  $CODE$ + 0006
;  229      0541  1 GLOBAL ROUTINE UPDATE_IBVBN (VBN) : NOVALUE =
;  230      0542  1
;  231      0543  1 !++
;  232      0544  1 !
;  233      0545  1 ! FUNCTIONAL DESCRIPTION:
;  234      0546  1 !
;  235      0547  1 !       This routine writes back the starting VBN for the index file bitmap
;  236      0548  1 !       scan into the vcb. This routine must be called in kernel mode.
;  237      0549  1 !
;  238      0550  1 ! CALLING SEQUENCE:
;  239      0551  1 !       UPDATE_VBN (ARG1)
;  240      0552  1 !
;  241      0553  1 ! INPUT PARAMETERS:
;  242      0554  1 !       ARG1: new start VBN
;  243      0555  1 !
;  244      0556  1 ! IMPLICIT INPUTS:
;  245      0557  1 !       CURRENT_VCB: VCB of volume
;  246      0558  1 !
;  247      0559  1 ! OUTPUT PARAMETERS:
;  248      0560  1 !       NONE
;  249      0561  1 !
;  250      0562  1 ! IMPLICIT OUTPUTS:
;  251      0563  1 !       NONE
;  252      0564  1 !
;  253      0565  1 ! ROUTINE VALUE:
;  254      0566  1 !       NONE
;  255      0567  1 !
;  256      0568  1 ! SIDE EFFECTS:
;  257      0569  1 !       VBN written into VCB
;  258      0570  1 !
;  259      0571  1 !--
;  260      0572  1
;  261      0573  2 BEGIN
;  262      0574  2
;  263      0575  2 EXTERNAL
;  264      0576  2         CURRENT_VCB     : REF BBLOCK;    ! VCB of volume
;  265      0577  2
;  266      0578  2 CURRENT_VCB[VCB$B_IBMAPVBN] = .VBN;
;  267      0579  2
;  268      0580  1 END;                                     ! end of routine UPDATE_VBN


                                      0000 00000          .ENTRY  UPDATE_IBVBN, Save nothing                   ; 0541
                   50      0000G  CF  D0 00002          MOVL    CURRENT_VCB, R0                              ; 0578
              3A   A0         04  AC  90 00007          MOVB    VBN, 58(R0)                                  ;
                                      04 0000C          RET                                                  ; 0580

; Routine Size:  13 bytes,    Routine Base:  $CODE$ + 0115
```

```
1 ROUTINE READ_NEW_HEADER (LBN) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine reads the block about to be used for a new file header.
1 !       It uses a local condition handler to fix up errors.
1 !
1 !
1 ! CALLING SEQUENCE:
1 !       READ_NEW_HEADER (ARG1)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: LBN of block to read
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       address of buffer containing block or 0 if bad
1 !
1 ! SIDE EFFECTS:
1 !       block read and/or written
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       HEADER          : REF BBLOCK;   ! address of block read
2
2 EXTERNAL ROUTINE
2       READ_BLOCK,                     ! read a block
2       WRITE_BLOCK,                    ! write a block
2       INVALIDATE,                     ! invalidate a buffer
2       CREATE_BLOCK;                   ! create a new block buffer
2
2 ! Under control of the condition handler, we read the block. If the read
2 ! fails, we attempt to rewrite the block and then read it again. If either
2 ! of the latter fails, we return failure.
2 !
2
2 ENABLE HANDLER;
2
2 HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
2
2 IF .HEADER EQL 0
2 THEN
3     BEGIN
3     HEADER = CREATE_BLOCK (.LBN, 1, HEADER_TYPE);
```

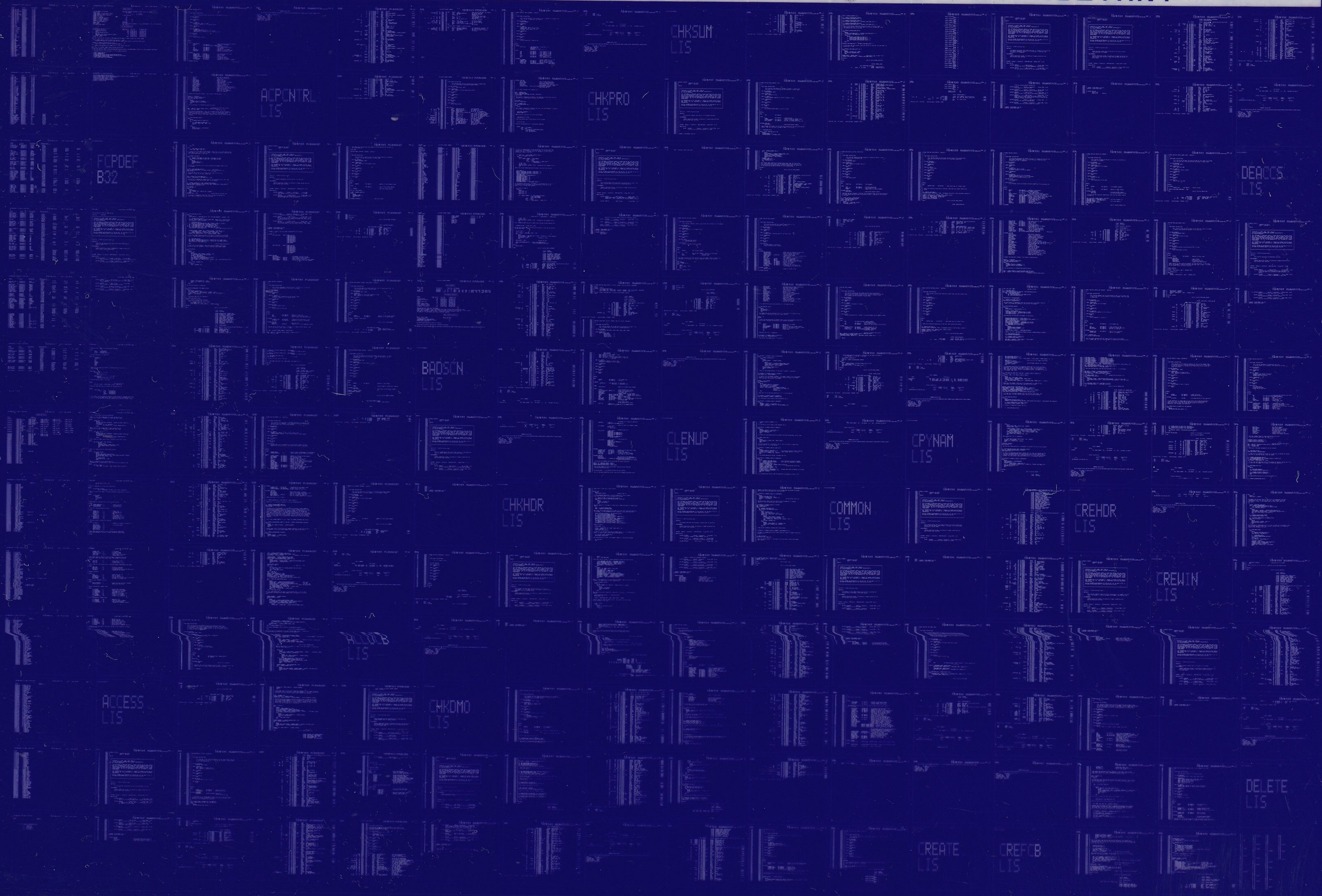
```
;  327          0638  3      WRITE_BLOCK (.HEADER);
;  328          0639  3      INVALIDATE (.HEADER);
;  329          0640  3      HEADER = READ_BLOCK (.LBN, 1, HEADER_TYPE);
;  330          0641  2      END;
;  331          0642  2
;  332          0643  2 RETURN .HEADER;
;  333          0644  2
;  334          0645  1 END;                                    ! end of routine READ_NEW_HEADER


                                                      .EXTRN  INVALIDATE, CREATE_BLOCK

                                      0004 00000 READ_NEW_HEADER:
                                                      .WORD   Save R2                                       ; 0581
                      6D     003F    CF   DE 00002    MOVAL   2$, (FP)                                      ; 0614
                      7E             01   7D 00007    MOVQ    #1, -(SP)                                     ; 0632
                              04     AC   DD 0000A    PUSHL   LBN                                           ;
                0000G CF             03   FB 0000D    CALLS   #3, READ_BLOCK                                ;
                      52             50   D0 00012    MOVL    R0, HEADER                                    ;
                                     2A   12 00015    BNEQ    1$                                            ; 0634
                      7E             01   7D 00017    MOVQ    #1, -(SP)                                     ; 0637
                              04     AC   DD 0001A    PUSHL   LBN                                           ;
                0000G CF             03   FB 0001D    CALLS   #3, CREATE_BLOCK                              ;
                      52             50   D0 00022    MOVL    R0, HEADER                                    ;
                                     52   DD 00025    PUSHL   HEADER                                        ; 0638
                0000G CF             01   FB 00027    CALLS   #1, WRITE_BLOCK                               ;
                                     52   DD 0002C    PUSHL   HEADER                                        ; 0639
                0000G CF             01   FB 0002E    CALLS   #1, INVALIDATE                                ;
                      7E             01   7D 00033    MOVQ    #1, -(SP)                                     ; 0640
                              04     AC   DD 00036    PUSHL   LBN                                           ;
                0000G CF             03   FB 00039    CALLS   #3, READ_BLOCK                                ;
                      52             50   D0 0003E    MOVL    R0, HEADER                                    ;
                      50             52   D0 00041 1$: MOVL   HEADER, R0                                    ; 0643
                                          04 00044    RET                                                   ; 0645
                                      0000 00045 2$:  .WORD   Save nothing                                  ; 0614
                                     7E   D4 00047    CLRL    -(SP)                                         ;
                                     5E   DD 00049    PUSHL   SP                                            ;
                      7E      04     AC   7D 0004B    MOVQ    4(AP), -(SP)                                  ;
                0000V CF             03   FB 0004F    CALLS   #3, HANDLER                                   ;
                                          04 00054    RET                                                   ;

; Routine Size:  85 bytes,    Routine Base:  $CODE$ + 0122
```

```
1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine is the condition handler for the initial header read.
1 !       On surface errors, it unwinds and causes a return of 0 to the caller
1 !       of the I/O routine to indicate error. Hard drive errors cause the
1 !       usual error exit.
1 !
1 ! CALLING SEQUENCE:
1 !       HANDLER (ARG1, ARG2)
1 !
1 ! INPUT PARAMETERS:
1 !       ARG1: address of signal array
1 !       ARG2: address of mechanism array
1 !
1 ! IMPLICIT INPUTS:
1 !       NONE
1 !
1 ! OUTPUT PARAMETERS:
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !       SS$_RESIGNAL or none if unwind
1 !
1 ! SIDE EFFECTS:
1 !       NONE
1 !
1 !--
1
1
2 BEGIN
2
2 MAP
2       SIGNAL          : REF BBLOCK,   ! signal arg array
2       MECHANISM       : REF BBLOCK;   ! mechanism arg array
2
2 EXTERNAL
2       IO_STATUS       : VECTOR;       ! I/O status block of last operation
2
2
2 ! If the condition is change mode to user (error exit) and the status is
2 ! read error, zero the return R0 and unwind to the the establisher. On
2 ! most write errors, zero the return R0 and unwind to the caller.
2 ! Otherwise, just resignal the condition.
2 !
2
2 IF .SIGNAL[CHF$L_SIG_NAME] EQL SS$_CMODUSER
2 THEN
3     BEGIN
3     MECHANISM[CHF$L_MCH_SAVR0] = 0;
3
```

```
;  393          0703  3       IF .SIGNAL[CHF$L_SIG_ARG1] EQL SS$_PARITY
;  394          0704  3       OR .SIGNAL[CHF$L_SIG_ARG1] EQL SS$_DATACHECK
;  395          0705  3       OR .SIGNAL[CHF$L_SIG_ARG1] EQL SS$_FORMAT
;  396          0706  3       THEN
;  397          0707  4          $UNWIND (DEPADR = MECHANISM[CHF$L_MCH_DEPTH])
;  398          0708  2       END;
;  399          0709  2
;  400          0710  2  RETURN SS$_RESIGNAL;                       ! status is irrelevant if unwinding
;  401          0711  2
;  402          0712  1  END;                                       ! end of routine HANDLER


                                                                   .EXTRN  IO_STATUS, SYS$UNWIND

                                          0000 00000 HANDLER:.WORD   Save nothing                          ; 0646
                         50         04  AC   D0 00002        MOVL    SIGNAL, R0                            ; 0698
              00000424   8F         04  A0   D1 00006        CMPL    4(R0), #1060                          ;
                                        31   12 0000E        BNEQ    2$                                    ;
                         51         08  AC   D0 00010        MOVL    MECHANISM, R1                         ; 0701
                                        0C   A1 D4 00014     CLRL    12(R1)                                ;
              000001F4   8F         08  A0   D1 00017        CMPL    8(R0), #500                           ; 0703
                                        14   13 0001F        BEQL    1$                                    ;
              0000005C   8F         08  A0   D1 00021        CMPL    8(R0), #92                            ; 0704
                                        0A   13 00029        BEQL    1$                                    ;
              000000BC   8F         08  A0   D1 0002B        CMPL    8(R0), #188                           ; 0705
                                        0C   12 00033        BNEQ    2$                                    ;
                                        7E   D4 00035 1$:    CLRL    -(SP)                                 ; 0707
                                    08  A1   9F 00037        PUSHAB  8(R1)                                 ;
              00000000G  00             02   FB 0003A        CALLS   #2, SYS$UNWIND                        ;
                         50       0918  8F   3C 00041 2$:    MOVZWL  #2328, R0                             ; 0710
                                             04 00046        RET                                           ; 0712

; Routine Size:  71 bytes,    Routine Base:  $CODE$ + 0177


;  403          0713  1
;  404          0714  1  END
;  405          0715  0  ELUDOM




;                                   PSECT SUMMARY
;
;       Name                     Bytes                      Attributes
;
;  $CODE$                            446  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                            Library Statistics
;
;                                       -------- Symbols --------      Pages      Processing
;      File                                Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1           18619       22         0       1000         00:01.8




;                                   COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LI[illegible]=LIS$:CREHDR/OBJ=OBJ$:CREHDR MSRC$:CREHDR/UPDATE=(ENH$:CREHDR)

; Size:          440 code + 6 data bytes
; Run Time:         00:12.0
; Elapsed Time:     00:36.4
; Lines/CPU Min:     3586
; Lexemes/CPU-Min: 13219
; Memory Used:  139 pages
; Comp[illegible] on Complete
```

0164 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



CHKSUM
LIS

ACPCNTRL
LIS

CHKPRO
LIS

FCPDEF
B32

DEACCS
LIS

BADSCN
LIS

CLENUP
LIS

CPYNAM
LIS

CHKHDR
LIS

COMMON
LIS

CREHDR
LIS

CREWIN
LIS

ALLOCB
LIS

ACCESS
LIS

CHKDMO
LIS

DELETE
LIS

CREATE
LIS

CREFCB
LIS