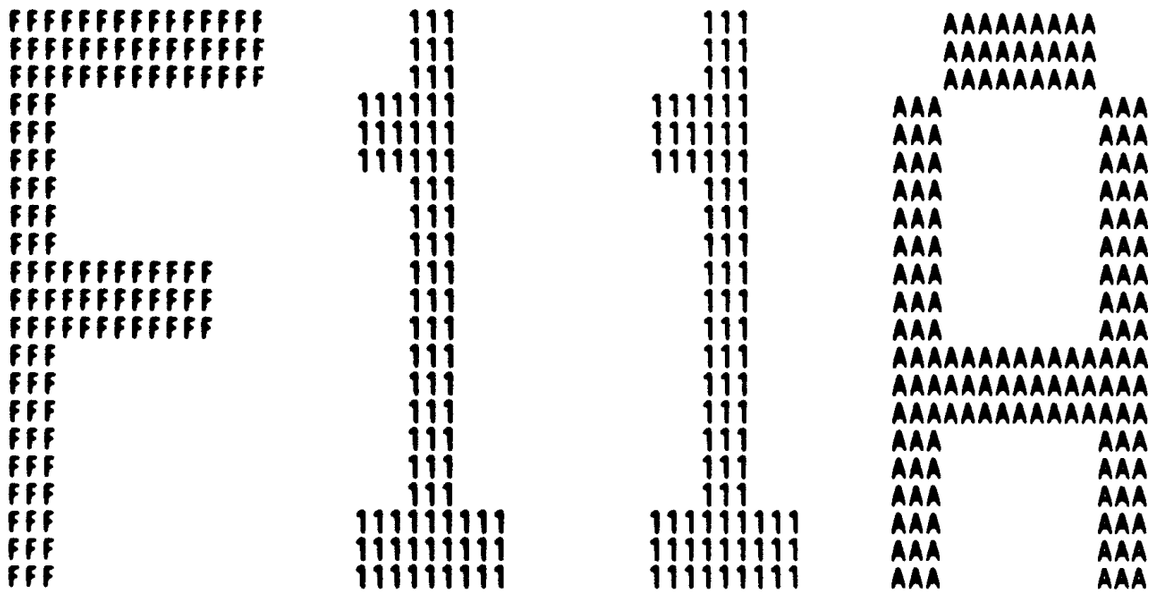
FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFFFFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFFFFFFFFFFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
FFF
111
111
111
111111
111111
111111
111
111
111
111
111
111
111
111
111
111
111111111
111111111
111111111
111
111
111
111111
111111
111111
111
111
111
111
111
111
111
111
111
111
111111111
111111111
111111111
AAAAAAAAA
AAAAAAAAA
AAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAAAAAAAAAAAAAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA
AAA AAA

```
WW      WW    IIIIII    TTTTTTTTTT  UU      UU  RRRRRRRR    NN      NN
WW      WW    IIIIII    TTTTTTTTTT  UU      UU  RRRRRRRR    NN      NN
WW      WW      II          TT      UU      UU  RR      RR  NN      NN
WW      WW      II          TT      UU      UU  RR      RR  NN      NN
WW      WW      II          TT      UU      UU  RR      RR  NNNN    NN
WW      WW      II          TT      UU      UU  RR      RR  NNNN    NN
WW      WW      II          TT      UU      UU  RRRRRRRR    NN  NN  NN
WW      WW      II          TT      UU      UU  RRRRRRRR    NN  NN  NN
WW  WW  WW      II          TT      UU      UU  RR  RR      NN    NNNN
WW  WW  WW      II          TT      UU      UU  RR  RR      NN    NNNN
WWWW  WWWW      II          TT      UU      UU  RR    RR    NN      NN      ....
WWWW  WWWW      II          TT      UU      UU  RR    RR    NN      NN      ....
WW      WW    IIIIII        TT      UUUUUUUUUU  RR      RR  NN      NN      ....
WW      WW    IIIIII        TT      UUUUUUUUUU  RR      RR  NN      NN      ....

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
```

```
;     1          0001  0 MODULE WITURN (
;     2          0002  0                 LANGUAGE (BLISS32),
;     3          0003  0                 IDENT = 'V04-000'
;     4          0004  0                 ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**************************************************************************
;     9          0009  1 !*                                                                        *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                   *
;    13          0013  1 !*                                                                        *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAI'ABLE TO ANY   *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    19          0019  1 !*  TRANSFERRED.                                                           *
;    20          0020  1 !*                                                                        *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    23          0023  1 !*  CORPORATION.                                                           *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
;    27          0027  1 !*                                                                        *
;    28          0028  1 !*                                                                        *
;    29          0029  1 !**************************************************************************
;    30          0030  1
;    31          0031  1 !++
;    32          0032  1 !
;    33          0033  1 ! FACILITY:  F11ACP Structure Level 1
;    34          0034  1 !
;    35          0035  1 ! ABSTRACT:
;    36          0036  1 !
;    37          0037  1 !       This module generates a window mapping the desired VBN from
;    38          0038  1 !       the supplied file header.
;    39          0039  1 !
;    40          0040  1 ! ENVIRONMENT:
;    41          0041  1 !
;    42          0042  1 !       STARLET operating system, including privileged system services
;    43          0043  1 !       and internal exec routines.
;    44          0044  1 !
;    45          0045  1 !--
;    46          0046  1 !
;    47          0047  1 !
;    48          0048  1 ! AUTHOR:  Andrew C. Goldstein, CREATION DATE:  7-Dec-1976  14:38
;    49          0049  1 !
;    50          0050  1 ! MODIFIED BY:
;    51          0051  1 !
;    52          0052  1 !       V03-005 LMP0254         L. Mark Pilant,         22-Jun-1984  15:16
;    53          0053  1 !               Don't clear the forward link of the window.  This was causing
;    54          0054  1 !               the ACP to accvio when doing a remap on cathedral windows.
;    55          0055  1 !
;    56          0056  1 !       V03-004 LMP0026         L. Mark Pilant,         18-May-1982  10:25
;    57          0057  1 !               Rearrange several instruction sequences to avoid possible
```

```
;    58          0058  1 !                 page faults at an elevated IPL.
;    59          0059  1 !
;    60          0060  1 !       V03-002 ACG0285         Andrew C. Goldstein,    12-Apr-1982  17:11
;    61          0061  1 !               Fix cathedral window logic for empty headers
;    62          0062  1 !
;    63          0063  1 !       V03-001 LMP0018         L. Mark Pilant,         31-Mar-1982  13:15
;    64          0064  1 !               Modify to use a local copy of the window complete flag.
;    65          0065  1 !               Also, fix the window building logic to correctly handle
;    66          0066  1 !               Cathedral windows as well as the garden variety windows.
;    67          0067  1 !
;    68          0068  1 !       V02-003 LMP0005         L. Mark Pilant,         29-Dec-1981  15:30
;    69          0069  1 !               Add support for Cathedral (segmented) windows.
;    70          0070  1 !
;    71          0071  1 !       V02-002 ACG0229         Andrew C. Goldstein,    22-Dec-1981  19:41
;    72          0072  1 !               Move updating of PMS$GL_TURN to MAP_VBN
;    73          0073  1 !
;    74          0074  1 !       V02-001 ACG0167         Andrew C. Goldstein,    7-May-1980  18:52
;    75          0075  1 !               Previous revision history moved to F11A.REV
;    76          0076  1 !**
;    77          0077  1
;    78          0078  1
;    79          0079  1 LIBRARY 'SYS$LIBRARY:LIB.L32';
;    80          0080  1 REQUIRE 'SRC$:FCPDEF.B32';
;    81          0395  1
;    82          0396  1 !
;    83          0397  1 ! The code in this routine must be locked into the working set because it
;    84          0398  1 ! runs at IPL$_SYNCH.
;    85          0399  1 !
;    86          0400  1
;    87          0401  1 LOCK_CODE;
```

```
;    89          0402  1 GLOBAL ROUTINE TURN_WINDOW (WINDOW, HEADER, DESIRED_VBN, START_VBN) =
;    90          0403  1
;    91          0404  1 !++
;    92          0405  1 !
;    93          0406  1 ! FUNCTIONAL DESCRIPTION:
;    94          0407  1 !
;    95          0408  1 !       This routine scans the map area of the supplied file header
;    96          0409  1 !       and builds retrieval pointers in the window until
;    97          0410  1 !       (1) the entire header has been scanned, or
;    98          0411  1 !       (2) the first retrieval pointer in the window maps the desired VBN
;    99          0412  1 !
;   100          0413  1 ! CALLING SEQUENCE:
;   101          0414  1 !       TURN_WINDOW (ARG1, ARG2, ARG3, ARG4)
;   102          0415  1 !
;   103          0416  1 ! INPUT PARAMETERS:
;   104          0417  1 !       ARG1: address of window block or 0 if to be created
;   105          0418  1 !       ARG2: address of file header
;   106          0419  1 !       ARG3: desired VBN
;   107          0420  1 !       ARG4: starting VBN of file header
;   108          0421  1 !
;   109          0422  1 ! IMPLICIT INPUTS:
;   110          0423  1 !       NONE
;   111          0424  1 !
;   112          0425  1 ! OUTPUT PARAMETERS:
;   113          0426  1 !       updated window
;   114          0427  1 !
;   115          0428  1 ! IMPLICIT OUTPUTS:
;   116          0429  1 !       NONE
;   117          0430  1 !
;   118          0431  1 ! ROUTINE VALUE:
;   119          0432  1 !       address of window if created
;   120          0433  1 !       or 1 if window already present
;   121          0434  1 !
;   122          0435  1 ! SIDE EFFECTS:
;   123          0436  1 !       NONE
;   124          0437  1 !
;   125          0438  1 !--
;   126          0439  1
;   127          0440  2 BEGIN
;   128          0441  2
;   129          0442  2 MAP
;   130          0443  2         WINDOW          : REF BBLOCK,   ! pointer to window
;   131          0444  2         HEADER          : REF BBLOCK;   ! pointer to file header
;   132          0445  2
;   133          0446  2 LABEL
;   134          0447  2         WINDOW_INIT;                    ! window initialization logic
;   135          0448  2
;   136          0449  2 LOCAL
;   137          0450  2         VBN,                            ! VBN in scanning window
;   138          0451  2         COUNT,                          ! retrieval pointer count
;   139          0452  2         COUNTER,                        ! loop counter
;   140          0453  2         LBN,                            ! retrieval pointer start LBN
;   141          0454  2         WINDOW_BUFFER   : BBLOCK [MAX_WINDOW*6], ! buffer in which to build window
;   142          0455  2         POINTER_COUNT,                  ! count of pointers in window
;   143          0456  2         WINDOW_SIZE,                    ! size of window in pointers
;   144          0457  2         BASE_VBN,                       ! starting VBN in window
;   145          0458  2         NEW_WINDOW      : REF BBLOCK,   ! address of newly allocated window
```

```
;  146     0459  2          W_POINTER       : REF BBLOCK,    ! pointer to scan window
;  147     0460  2          M_POINTER       : REF BBLOCK,    ! pointer to scan header map area
;  148     0461  2          MAP_AREA        : REF BBLOCK,    ! pointer to start of header map area
;  149     0462  2          MAP_POINTER     : REF BBLOCK,    ! pointer to scan header map area
;  150     0463  2          WINDOW_COUNT,                    ! number of blocks mapped by the window
;  151     0464  2          PRIMARY_WINDOW  : REF BBLOCK;    ! address of the primary window segment
;  152     0465  2
;  153     0466  2 MACRO
;  154     0467  2          WINDOW_MAP      = (.WINDOW+WCB$C_MAP)%; !start of window map area
;  155     0468  2
;  156     0469  2 EXTERNAL
;  157     0470  2          CLEANUP_FLAGS   : BITVECTOR;     ! cleanup action and status flags
;  158     0471  2
;  159     0472  2 EXTERNAL ROUTINE
;  160     0473  2          ALLOCATE;                        ! allocate system dynamic memory
;  161     0474  2
;  162     0475  2 ! There are two general cases that can occur upon entering the window turner:
;  163     0476  2 ! 1) a window previously exists, in which case things get a little messy; or
;  164     0477  2 ! 2) no window previously exists, in which case things are very simply.                    ; Rc
;  165     0478  2 !
;  166     0479  2
;  167     0480  2 W_POINTER = WINDOW_BUFFER;
;  168     0481  2 WINDOW_COUNT = 0;
;  169     0482  2
;  170     0483  2 M_POINTER = 0;
;  171     0484  2 MAP_AREA = .HEADER + .HEADER[FH1$B_MPOFFSET] * 2;        ! point to map area
;  172     0485  2 MAP_POINTER = .MAP_AREA + FM1$C_POINTERS;                ! point to start of pointers
;  173     0486  2
;  174     0487  2 IF .WINDOW EQL 0
;  175     0488  2 THEN
;  176     0489  2
;  177     0490  2 ! Create a new window.  All that is necessary is to initialize a few variables
;  178     0491  2 ! and pointers and then start reading the file header.
;  179     0492  2 !
;  180     0493  2
;  181     0494  3     BEGIN
;  182     0495  3     BASE_VBN = .START_VBN;
;  183     0496  3     WINDOW_SIZE = MAX_WINDOW;
;  184     0497  3     PRIMARY_WINDOW = NEW_WINDOW = 0;
;  185     0498  3     POINTER_COUNT = 0;
;  186     0499  3     END
;  187     0500  2 ELSE
;  188     0501  2
;  189     0502  2 ! Use an existing window.  Several situation may occur in this case: 1) the
;  190     0503  2 ! window must be turned to point to a different portion of the file; 2) the
;  191     0504  2 ! header contains pointers which may be added after truncating the existing
;  192     0505  2 ! window, this usually occurs when a file is extended without causing a new
;  193     0506  2 ! file header to be created; 3) the desired VBN is less than the specifed
;  194     0507  2 ! starting VBN and the starting VBN is greater than 1, this occurs when a
;  195     0508  2 ! file is extended and a new file header had to be created; or 4) the window
;  196     0509  2 ! already maps a portion of the header and only the new pointers (which may
;  197     0510  2 ! include a partial map pointer if two contiguous extents were collapsed into
;  198     0511  2 ! one map pointer in the header).
;  199     0512  2 !
;  200     0513  2
;  201     0514  3     WINDOW_INIT: BEGIN
;  202     0515  3     BASE_VBN = .WINDOW[WCB$L_STVBN];
```

```
;  203          0516  3     WINDOW_SIZE = MINU ((.WINDOW[WCB$W_SIZE] - WCB$C_LENGTH) / 6, MAX_WINDOW);
;  204          0517  3     PRIMARY_WINDOW = NEW_WINDOW = .WINDOW;
;  205          0518  3     POINTER_COUNT = .WINDOW[WCB$W_NMAP];
;  206          0519  3     CH$MOVE (.POINTER_COUNT * 6, WINDOW_MAP, WINDOW_BUFFER);     ! copy current
;  207          0520  3     VBN = .BASE_VBN;
;  208          0521  3
;  209          0522  3     IF .START_VBN LEQU .VBN
;  210          0523  3     OR
;  211          0524  3
;  212          0525  3 ! Determine if the existing window may be truncated.
;  213          0526  3 !
;  214          0527  3
;  215          0528  4         BEGIN
;  216          0529  4         INCR J FROM 1 TO .POINTER_COUNT
;  217          0530  4         DO
;  218          0531  5             BEGIN
;  219          0532  5             VBN = .VBN + .W_POINTER[WCB$W_COUNT];        ! VBN at the end of the pointer
;  220          0533  5             WINDOW_COUNT = .WINDOW_COUNT + .W_POINTER[WCB$W_COUNT];
;  221          0534  5             W_POINTER = .W_POINTER + 6;
;  222          0535  5             IF .START_VBN LEQ .VBN
;  223          0536  5             THEN
;  224          0537  6                 BEGIN
;  225          0538  6                 W_POINTER[WCB$W_PREVCOUNT] = .W_POINTER[WCB$W_PREVCOUNT]
;  226          0539  6                                            - (.VBN - .START_VBN);
;  227          0540  6                 POINTER_COUNT = .J;                          ! note where window truncated
;  228          0541  6                 LEAVE WINDOW_INIT;
;  229          0542  5                 END;
;  230          0543  4             END;
;  231          0544  4         NOT .WINDOW[WCB$V_CATHEDRAL]
;  232          0545  4         END
;  233          0546  3     THEN
;  234          0547  3
;  235          0548  3 ! Either the window cannot be truncated or the header maps before the beginning
;  236          0549  3 ! of the existing window.  In the latter case if cathedral windows are not in
;  237          0550  3 ! use the window may be discarded; if cathedral windows are in use then it is
;  238          0551  3 ! possible that a beginning portion of the header may be discarded.
;  239          0552  3 !
;  240          0553  4         BEGIN
;  241          0554  4         IF .WINDOW[WCB$V_CATHEDRAL]
;  242          0555  4         THEN
;  243          0556  5             BEGIN
;  244          0557  5             IF .BASE_VBN EQL .START_VBN
;  245          0558  5             THEN
;  246          0559  6                 BEGIN
;  247          0560  6                 POINTER_COUNT = 0;
;  248          0561  6                 LEAVE WINDOW_INIT;
;  249          0562  5                 END;
;  250          0563  5             IF
;  251          0564  6                 BEGIN
;  252          0565  6                 VBN = .START_VBN;
;  253          0566  6                 DECR J FROM .MAP_AREA[FM1$B_INUSE]/2 TO 1
;  254          0567  6                 DO
;  255          0568  7                     BEGIN
;  256          0569  7                     M_POINTER = .MAP_POINTER;
;  257          0570  7                     COUNT = .MAP_POINTER[FM1$B_COUNT] + 1;
;  258          0571  7                     MAP_POINTER = .MAP_POINTER + 4;      ! update map pointer
;  259          0572  7                     IF .BASE_VBN GEQ .VBN
```

```
;  260          0573  7                    AND .BASE_VBN LSS .VBN + .COUNT
;  261          0574  7                    THEN
;  262          0575  8                        BEGIN
;  263          0576  8                        MAP_POINTER = .M_POINTER;       ! back up the header pointer
;  264          0577  8                        POINTER_COUNT = 0;
;  265          0578  8                        LEAVE WINDOW_INIT;
;  266          0579  7                        END;
;  267          0580  7                    VBN = .VBN + .COUNT;
;  268          0581  7                    END
;  269          0582  6                END
;  270          0583  5            THEN BUG_CHECK (WCBFCBMNG, FATAL, 'WCB/FCB correspondence broken');
;  271          0584  5            END
;  272          0585  4        ELSE
;  273          0586  5            BEGIN
;  274          0587  5            IF .VBN EQLU .BASE_VBN
;  275          0588  5            AND .DESIRED_VBN LSSU .START_VBN
;  276          0589  5            AND .START_VBN GTRU 1
;  277          0590  5            THEN RETURN 1;                          ! note window already best fit
;  278          0591  5
;  279          0592  5 ! The header maps before the existing window and cathedral windows are not in
;  280          0593  5 ! use.  Discard the existing window and start a new one.
;  281          0594  5 !
;  282          0595  5
;  283          0596  5            POINTER_COUNT = 0;
;  284          0597  5            WINDOW_COUNT = 0;
;  285          0598  5            BASE_VBN = .START_VBN;
;  286          0599  5            M_POINTER = WINDOW_BUFFER;
;  287          0600  4            END;
;  288          0601  3        END;
;  289          0602  2    END;                                            ! end of block WINDOW_INIT
;  290          0603  2
;  291          0604  2 ! The window is now suitably initialized. Set up necessary pointers.
;  292          0605  2 ! Now scan the map area, extracting retrieval pointers.
;  293          0606  2 !
;  294          0607  2
;  295          0608  2 UNTIL .MAP_POINTER GEQA .MAP_AREA + FM1$C_POINTERS + .MAP_AREA[FM1$B_INUSE] * 2
;  296          0609  2 DO
;  297          0610  3     BEGIN
;  298          0611  3
;  299          0612  3     COUNT = .MAP_POINTER[FM1$B_COUNT] + 1;
;  300          0613  3     LBN = .MAP_POINTER[FM1$W_LOWLBN];
;  301          0614  3     LBN<16,8> = .MAP_POINTER[FM1$B_HIGHLBN];
;  302          0615  3     MAP_POINTER = .MAP_POINTER + 4;
;  303          0616  3
;  304          0617  3 ! If the existing window's first map pointer is part of a set needed to map
;  305          0618  3 ! a map pointer from the header (this only occurs if the map pointer from the
;  306          0619  3 ! file header maps more than 65535 blocks), it is necessary to adjust the
;  307          0620  3 ! block count and LBN returned from the header.
;  308          0621  3 !
;  309          0622  3
;  310          0623  3     IF .M_POINTER NEQ 0
;  311          0624  3     THEN
;  312          0625  4         BEGIN
;  313          0626  4         COUNT = .COUNT - (.BASE_VBN - .VBN);
;  314          0627  4         LBN = .LBN + (.BASE_VBN - .VBN);
;  315          0628  4         M_POINTER = 0;                              ! only needed once
;  316          0629  3         END;
```

```
;  317     0630  3
;  318     0631  3 ! Build new retrieval pointers, using as many as needed to run out the
;  319     0632  3 ! count. If the window is full, and segmented windows are not allowed,
;  320     0633  3 ! shuffle the entries up by one. If this would cause the pointer mapping
;  321     0634  3 ! the desired VBN to fall off the top, we are done. If segmented windows
;  322     0635  3 ! are allowed, write out the current segment, allocate room for the next
;  323     0636  3 ! segment, reset the critical counts and pointers, and continue on my
;  324     0637  3 ! merry way.
;  325     0638  3 !
;  326     0639  3
;  327     0640  3     IF .COUNT NEQ 0
;  328     0641  3     THEN
;  329     0642  4         BEGIN
;  330     0643  4
;  331     0644  4 ! If the current set of blocks mapped is contiguous with the preceeding set
;  332     0645  4 ! (in the window) and the total count is less than the maximum allowed, merge
;  333     0646  4 ! the two sets.  Otherwise, create a new window map pointer.
;  334     0647  4 !
;  335     0648  4
;  336     0649  4         IF .POINTER_COUNT NEQ 0                    ! only for non-empty window
;  337     0650  4         AND .W_POINTER[WCB$W_PREVCOUNT] + .W_POINTER[WCB$L_PREVLBN] EQL .LBN
;  338     0651  4         AND .W_POINTER[WCB$W_PREVCOUNT] + .COUNT LSSU 65536
;  339     0652  4         THEN W_POINTER[WCB$W_PREVCOUNT] = .W_POINTER[WCB$W_PREVCOUNT] + .COUNT
;  340     0653  4         ELSE
;  341     0654  5             BEGIN
;  342     0655  5             IF .POINTER_COUNT GEQU .WINDOW_SIZE
;  343     0656  5             THEN
;  344     0657  6                 BEGIN
;  345     0658  6                 IF .WINDOW NEQ 0 AND NOT .WINDOW[WCB$V_CATHEDRAL]
;  346     0659  6                 THEN
;  347     0660  7                     BEGIN
;  348     0661  7                     CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
;  349     0662  7                     IF .BASE_VBN + .WINDOW_BUFFER[WCB$W_COUNT] GTRU .DESIRED_VBN
;  350     0663  7                     THEN EXITLOOP;
;  351     0664  7
;  352     0665  7                     POINTER_COUNT = .POINTER_COUNT - 1;
;  353     0666  7                     BASE_VBN = .BASE_VBN + .WINDOW_BUFFER[WCB$W_COUNT];
;  354     0667  7                     CH$MOVE (.POINTER_COUNT*6, WINDOW_BUFFER+6, WINDOW_BUFFER);
;  355     0668  7                     W_POINTER = .W_POINTER - 6;
;  356     0669  7                     END
;  357     0670  6                 ELSE
;  358     0671  7                     BEGIN
;  359     0672  7                     LOCAL       TEMP_LINK       : REF BBLOCK;   ! addr of temp window link
;  360     0673  7                     IF .PRIMARY_WINDOW EQL 0
;  361     0674  7                     THEN
;  362     0675  8                         BEGIN
;  363     0676  9                         PRIMARY_WINDOW = NEW_WINDOW = ALLOCATE ((MAXU (.POINTER_COUNT, MIN_WINDOW))
;  364     0677  8                                                                 * 6 + WCB$C_LENGTH, WCB_TYPE);
;  365     0678  8                         IF .NEW_WINDOW EQL 0
;  366     0679  8                         THEN
;  367     0680  9                             BEGIN
;  368     0681  9                             CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
;  369     0682  9                             RETURN 0;
;  370     0683  8                             END;
;  371     0684  7                         END;
;  372     0685  7                     IF .NEW_WINDOW[WCB$L_LINK] NEQ 0
;  373     0686  7                     THEN TEMP_LINK = .NEW_WINDOW[WCB$L_LINK]
```

```
;  374     0687  7                    ELSE
;  375     0688  8                        BEGIN
;  376     0689  8                        WINDOW_SIZE = MAX_WINDOW;
;  377     0690  8                        TEMP_LINK = ALLOCATE (.WINDOW_SIZE * 6 + WCB$C_LENGTH, WCB_TYPE);
;  378     0691  8                        IF .TEMP_LINK EQL 0
;  379     0692  8                        THEN
;  380     0693  9                            BEGIN
;  381     0694  9                            CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
;  382     0695  9                            EXITLOOP;
;  383     0696  8                            END;
;  384     0697  7                        END;
;  385     0698  7                    SET_IPL (;PL$_SYNCH);
;  386     0699  7
;  387     0700  7 ! Copy the needed information from the previous window segment.
;  388     0701  7 !
;  389     0702  7
;  390     0703  7                    TEMP_LINK[WCB$L_PID] = .NEW_WINDOW[WCB$L_PID];
;  391     0704  7                    TEMP_LINK[WCB$L_ORGUCB] = .NEW_WINDOW[WCB$L_ORGUCB];
;  392     0705  7                    TEMP_LINK[WCB$W_ACON] = .NEW_WINDOW[WCB$W_ACON];
;  393     0706  7                    TEMP_LINK[WCB$L_FCB] = .NEW_WINDOW[WCB$L_FCB];
;  394     0707  7                    TEMP_LINK[WCB$B_ACCESS] = .NEW_WINDOW[WCB$B_ACCESS];
;  395     0708  7                    TEMP_LINK[WCB$L_RVT] = .NEW_WINDOW[WCB$L_RVT];
;  396     0709  7
;  397     0710  7 ! Finish up the current segment and create another.
;  398     0711  7 !
;  399     0712  7
;  400     0713  7                    NEW_WINDOW[WCB$L_LINK] = .TEMP_LINK;
;  401     0714  7                    NEW_WINDOW[WCB$V_CATHEDRAL] = T;
;  402     0715  7                    NEW_WINDOW[WCB$W_NMAP] = .POINTER_COUNT;
;  403     0716  7                    NEW_WINDOW[WCB$L_STVBN] = .BASE_VBN;
;  404     0717  7                    CH$MOVE (.POINTER_COUNT*6, WINDOW_BUFFER, .NEW_WINDOW+WCB$C_MAP);
;  405     0718  7                    NEW_WINDOW = .NEW_WINDOW[WCB$L_LINK];
;  406     0719  7                    BASE_VBN = .BASE_VBN + .WINDOW_COUNT;
;  407     0720  7                    NEW_WINDOW[WCB$W_NMAP] = 0;
;  408     0721  7                    NEW_WINDOW[WCB$L_STVBN] = .BASE_VBN;
;  409     0722  7                    NEW_WINDOW[WCB$V_CATHEDRAL] = 1;
;  410     0723  7                    SET_IPL (0);
;  411     0724  7                    W_POINTER = WINDOW_BUFFER;
;  412     0725  7                    POINTER_COUNT = 0;
;  413     0726  7                    WINDOW_COUNT = 0;
;  414     0727  6                    END;
;  415     0728  5                END;
;  416     0729  5
;  417     0730  5 ! Finally build the pointer and count it.
;  418     0731  5 !
;  419     0732  5
;  420     0733  5            W_POINTER[WCB$W_COUNT] = .COUNT;
;  421     0734  5            WINDOW_COUNT = .WINDOW_COUNT + .W_POINTER[WCB$W_COUNT];
;  422     0735  5            W_POINTER[WCB$L_LBN] = .LBN;
;  423     0736  5            W_POINTER = .W_POINTER + 6;
;  424     0737  5            POINTER_COUNT = .POINTER_COUNT + 1;
;  425     0738  4            END;
;  426     0739  3        END;
;  427     0740  3
;  428     0741  2    END;                                          ! end of header scan loop
;  429     0742  2
;  430     0743  2 ! Having built a new window in the buffer, update the real one. Then interlock
```

```
;  431          0744  2 ! the system data base and copy the buffer into the window.
;  432          0745  2 !
;  433          0746  2
;  434          0747  2 IF .NEW_WINDOW EQL 0
;  435          0748  2 THEN
;  436          0749  3     BEGIN
;  437          0750  3     IF .WINDOW EQL 0
;  438          0751  3     THEN
;  439          0752  4         BEGIN
;  440          0753  5         PRIMARY_WINDOW =  NEW_WINDOW = ALLOCATE ((MAXU (.POINTER_COUNT, MIN_WINDOW))
;  441          0754  4                                                  * 6 + WCB$C_LENGTH, WCB_TYPE);
;  442          0755  4         IF .NEW_WINDOW EQL 0
;  443          0756  4         THEN
;  444          0757  5             BEGIN
;  445          0758  5             CLEANUP_FLAGS[CLF_INCOMPLETE] = 1;
;  446          0759  5             RETURN 0;
;  447          0760  4             END;
;  448          0761  4         NEW_WINDOW[WCB$V_CATHEDRAL] = 1;
;  449          0762  4         END
;  450          0763  3     ELSE NEW_WINDOW = .WINDOW;
;  451          0764  2     END;
;  452          0765  2
;  453          0766  2 SET_IPL (IPL$_SYNCH);
;  454          0767  2 NEW_WINDOW[WCB$W_NMAP] = .POINTER_COUNT;
;  455          0768  2 NEW_WINDOW[WCB$L_STVBN] = .BASE_VBN;
;  456          0769  2 CH$MOVE (.POINTER_COUNT*6, WINDOW_BUFFER, .NEW_WINDOW+WCB$C_MAP);
;  457          0770  2 SET_IPL (0);                                    ! unlock the data base
;  458          0771  2
;  459          0772  2 IF .WINDOW EQL 0
;  460          0773  2 THEN RETURN .PRIMARY_WINDOW
;  461          0774  2 ELSE RETURN 1;
;  462          0775  2
;  463          0776  1 END;                                            ! end of routine TURN_WINDOW


                                                          .TITLE  WITURN
                                                          .IDENT  \V04-000\

                                                          .EXTRN  CLEANUP_FLAGS, ALLOCATE
                                                          .EXTRN  BUG$_WCBFCBMNG

                                                          .PSECT  $LOCKEDC1$,NOWRT,2

                                        0FFC 00000        .ENTRY  TURN_WINDOW, Save R2,R3,R4,R5,R6,R7,R8,R9,- ; 0402
                                                                  R10,R11
                                    5E  FE00   CE  9E 00002 MOVAB   -512(SP), SP
                                    56    20   AE  9E 00007 MOVAB   WINDOW_BUFFER, W_POINTER                  ; 0480
                                          0C   AE  7C 0000B CLRQ    M_POINTER                               ; 0483
                                    51    08   AC  D0 0000E MOVL    HEADER, R1                              ; 0484
                                    50    01   A1  9A 00012 MOVZBL  1(R1), R0
                                    5A         6140  3E 00016 MOVAW   (R1)[R0], MAP_AREA
                                    57    0A   AA  9E 0001A MOVAB   10(R10), MAP_POINTER                    ; 0485
                                          04   AC  D5 0001E TSTL    WINDOW                                  ; 0487
                                               0E  12 00021 BNEQ    1$
                                    59    10   AC  D0 00023 MOVL    START_VBN, BASE_VBN                     ; 0495
                                08  AE    50   8F  9A 00027 MOVZBL  #80, WINDOW_SIZE                        ; 0496
                                               6E  7C 0002C CLRQ    PRIMARY_WINDOW                          ; 0497
```

```
                                    00C1 31 0002E       BRW     7$                                ; 0498
      50     04  AC            2C C1 00031 1$:  ADDL3   #44, WINDOW, R0                   ; 0515
                 59            60 D0 00036      MOVL    (R0), BASE_VBN                    ;
      51     04  AC            08 C1 00039      ADDL3   #8, WINDOW, R1                    ; 0516
                 50            61 3C 0003E      MOVZWL  (R1), R0                          ;
                 50            30 C2 00041      SUBL2   #48, R0                           ;
                 50            06 C6 00044      DIVL2   #6, R0                            ;
       00000050  8F            50 D1 00047      CMPL    R0, #80                           ;
                               04 1B 0004E      BLEQU   2$                                ;
                 50     50     8F 9A 00050      MOVZBL  #80, R0                           ;
             08  AE            50 D0 00054 2$:  MOVL    R0, WINDOW_SIZE                   ;
             04  AE     04     AC D0 00058      MOVL    WINDOW, NEW_WINDOW                ; 0517
                 6E     04     AC D0 0005D      MOVL    WINDOW, PRIMARY_WINDOW            ;
      50     04  AC            16 C1 00061      ADDL3   #22, WINDOW, R0                   ; 0518
                 58            60 3C 00066      MOVZWL  (R0), POINTER_COUNT               ;
      50         58            06 C5 00069      MULL3   #6, R8, R0                        ; 0519
      5B     04  AC            30 C1 0006D      ADDL3   #48, WINDOW, R11                  ;
 20   AE         6B            50 28 00072      MOVC3   R0, (R11), WINDOW_BUFFER          ;
                 5B            59 D0 00077      MOVL    BASE_VBN, VBN                     ; 0520
                 5B     10     AC D1 0007A      CMPL    START_VBN, VBN                    ; 0522
                               35 1B 0007E      BLEQU   5$                                ;
                               51 D4 00080      CLRL    J                                 ; 0529
                               24 11 00082      BRB     4$                                ;
                 50            66 3C 00084 3$:  MOVZWL  (W_POINTER), R0                   ; 0532
                 5B            50 C0 00087      ADDL2   R0, VBN                           ;
                 50            86 3C 0008A      MOVZWL  (W_POINTER)+, R0                  ; 0533
             10  AE            50 C0 0008D      ADDL2   R0, WINDOW_COUNT                  ;
                 56            04 C0 00091      ADDL2   #4, W_POINTER                     ; 0534
                 5B     10     AC D1 00094      CMPL    START_VBN, VBN                    ; 0535
                               0E 14 00098      BGTR    4$                                ;
      50     10  AC            5B C3 0009A      SUBL3   VBN, START_VBN, R0                ; 0539
             FA  A6            50 A0 0009F      ADDW2   R0, -6(W_POINTER)                 ;
                 58            51 D0 000A3      MOVL    J, POINTER_COUNT                  ; 0540
                               7D 11 000A6      BRB     12$                               ; 0541
      D8         51            58 F3 000A8 4$:  AOBLEQ  R8, J, 3$                         ; 0529
      50     04  AC            0B C1 000AC      ADDL3   #11, WINDOW, R0                   ; 0544
      70         60            06 E0 000B1      BBS     #6, (R0), 12$                     ;
      52     04  AC            0B C1 000B5 5$:  ADDL3   #11, WINDOW, R2                   ; 0554
      45         62            06 E1 000BA      BBC     #6, (R2), 10$                     ;
             10  AC            59 D1 000BE      CMPL    BASE_VBN, START_VBN               ; 0557
                               2E 13 000C2      BEQL    7$                                ;
                 5B     10     AC D0 000C4      MOVL    START_VBN, VBN                    ; 0565
                 51     08     AA 9A 000C8      MOVZBL  8(MAP_AREA), R1                   ; 0566
                 51            02 C6 000CC      DIVL2   #2, R1                            ;
                               51 D6 000CF      INCL    J                                 ;
                               27 11 000D1      BRB     9$                                ;
             0C  AE            87 DE 000D3 6$:  MOVAL   (MAP_POINTER)+, M_POINTER         ; 0569
             14  AE     FD     A7 9A 000D7      MOVZBL  -3(MAP_POINTER), COUNT            ; 0570
                        14     AE D6 000DC      INCL    COUNT                             ;
                 5B            59 D1 000DF      CMPL    BASE_VBN, VBN                     ; 0572
                               12 19 000E2      BLSS    8$                                ;
      50         5B     14     AE C1 000E4      ADDL3   COUNT, VBN, R0                    ; 0573
                 50            59 D1 000E9      CMPL    BASE_VBN, R0                      ;
                               08 18 000EC      BGEQ    8$                                ;
                 57     0C     AE D0 000EE      MOVL    M_POINTER, MAP_POINTER            ; 0576
                               58 D4 000F2 7$:  CLRL    POINTER_COUNT                     ; 0577
                               2F 11 000F4      BRB     12$                               ; 0578
```

```
                5B      14   AE  C0 000F6 8$:      ADDL2   COUNT, VBN                                  ; 0580
                D6           51  F5 000FA 9$:      SOBGTR  R1, 6$                                      ; 0566
                               FEFF 000FD          BUGW                                                ; 0583
                              0000* 000FF          .WORD   <BUG$_WCBFCBMNG!4>
                              22  11 00101        BRB     12$                                         ; 0554
                59                 5B  D1 00103 10$:     CMPL    VBN, BASE_VBN                       ; 0587
                                   10  12 00106         BNEQ    11$
              10    AC        0C   AC  D1 00108         CMPL    DESIRED_VBN, START_VBN              ; 0588
                                   09  1E 0010D         BGEQU   11$
                01          10   AC  D1 0010F         CMPL    START_VBN, #1                       ; 0589
                                   03  1B 00113         BLEQU   11$
                                0228  31 00115         BRW     31$
                                   58  D4 00118 11$:     CLRL    POINTER_COUNT                       ; 0596
                              10   AE  D4 0011A         CLRL    WINDOW_COUNT                        ; 0597
                59            10   AC  D0 0011D         MOVL    START_VBN, BASE_VBN                 ; 0598
                56            20   AE  9E 00121         MOVAB   WINDOW_BUFFER, W_POINTER            ; 0599
              1C    AE        08   AA  9E 00125 12$:     MOVAB   8(MAP_AREA), 28(SP)                 ; 0608
                50            1C   BE  9A 0012A 13$:     MOVZBL  @28(SP), R0
                50          0A AA40  3E 0012E         MOVAW   10(MAP_AREA)[R0], R0
                50                 57  D1 00133         CMPL    MAP_POINTER, R0
                                   03  1F 00136         BLSSU   14$
                                0199  31 00138         BRW     25$
              14    AE        01   A7  9A 0013B 14$:     MOVZBL  1(MAP_POINTER), COUNT               ; 0612
                              14   AE  D6 00140         INCL    COUNT
              18    AE        02   A7  3C 00143         MOVZWL  2(MAP_POINTER), LBN                 ; 0613
              1A    AE              87  90 00148         MOVB    (MAP_POINTER)+, LBN+2               ; 0614
                57                 03  C0 0014C         ADDL2   #3, MAP_POINTER                     ; 0615
                              0C   AE  D5 0014F         TSTL    M_POINTER                           ; 0623
                                   0F  13 00152         BEQL    15$
    50          5B                 59  C3 00154         SUBL3   BASE_VBN, VBN, R0                   ; 0626
              14    AE              50  C0 00158         ADDL2   R0, COUNT
              18    AE              50  C2 0015C         SUBL2   R0, LBN                             ; 0627
                              0C   AE  D4 00160         CLRL    M_POINTER                           ; 0628
                              14   AE  D5 00163 15$:     TSTL    COUNT                               ; 0640
                                   C2  13 00166         BEQL    13$
                                   58  D5 00168         TSTL    POINTER_COUNT                       ; 0649
                                   26  13 0016A         BEQL    16$
                50          FA   A6  3C 0016C         MOVZWL  -6(W_POINTER), R0                   ; 0650
                50          FC   A6  C0 00170         ADDL2   -4(W_POINTER), R0
              18    AE              50  D1 00174         CMPL    R0, LBN
                                   18  12 00178         BNEQ    16$
                50          FA   A6  3C 0017A         MOVZWL  -6(W_POINTER), R0                   ; 0651
                50          14   AE  C0 0017E         ADDL2   COUNT, R0
          00010000          8F       50  D1 00182         CMPL    R0, #65536
                                   07  1E 00189         BGEQU   16$
              FA    A6        14   AE  A0 0018B         ADDW2   COUNT, -6(W_POINTER)                ; 0652
                                   98  11 00190         BRB     13$
              08    AE              58  D1 00192 16$:     CMPL    POINTER_COUNT, WINDOW_SIZE          ; 0655
                                   33  1F 00196         BLSSU   17$
                50          04   AC  D0 00198         MOVL    WINDOW, R0                          ; 0658
                                   30  13 0019C         BEQL    18$
  2B          0B    A0             06  E0 0019E         BBS     #6, 11(R0), 18$
            0000G   CF             04  88 001A3         BISB2   #4, CLEANUP_FLAGS+1                 ; 0661
                50          20   AE  3C 001A8         MOVZWL  WINDOW_BUFFER, R0                   ; 0662
                50                 59  C0 001AC         ADDL2   BASE_VBN, R0
              0C    AC              50  D1 001AF         CMPL    R0, DESIRED_VBN
                                   6F  1A 001B3         BGTRU   22$
```

```
                                                58  D7 001B5          DECL    POINTER_COUNT                                   ; 0665
                                            50  20  AE 3C 001B7          MOVZWL  WINDOW_BUFFER, R0                       ; 0666
                                            59      50 C0 001BB          ADDL2   R0, BASE_VBN                            ;
                                        50          58 06 C5 001BE          MULL3   #6, POINTER_COUNT, R0               ; 0667
                                    20  AE      26  AE 50 28 001C2          MOVC3   R0, WINDOW_BUFFER+6, WINDOW_BUFFER  ;
                                            56      06 C2 001C8          SUBL2   #6, W_POINTER                           ; 0668
                                                00F2 31 001CB 17$:      BRW     24$                                     ; 0658
                                                  6E D5 001CE 18$:      TSTL    PRIMARY_WINDOW                          ; 0673
                                                  22 12 001D0           BNEQ    20$                                     ;
                                                  01 DD 001D2           PUSHL   #1                                      ; 0676
                                            50        58 D0 001D4           MOVL    POINTER_COUNT, R0                     ;
                                                      03 12 001D7           BNEQ    19$                                   ;
                                            50        01 D0 001D9           MOVL    #1, R0                                ;
                                            50        06 C4 001DC 19$:      MULL2   #6, R0                                ; 0677
                                        30  A0    9F 001DF              PUSHAB  48(R0)                                  ;
                                    0000G   CF        02 FB 001E2           CALLS   #2, ALLOCATE                          ;
                                      04    AE        50 D0 001E7           MOVL    R0, NEW_WINDOW                        ;
                                            6E  04    AE D0 001EB           MOVL    NEW_WINDOW, PRIMARY_WINDOW            ; 0676
                                                      03 12 001EF           BNEQ    20$                                   ; 0678
                                                  0109 31 001F1           BRW     27$                                   ;
                                        50      04    AE    20 C1 001F4 20$:  ADDL3   #32, NEW_WINDOW, R0                 ; 0685
                                                            60 D5 001F9       TSTL    (R0)                                ;
                                                            0A 13 001FB       BEQL    21$                                 ;
                                        51      04    AE    20 C1 001FD       ADDL3   #32, NEW_WINDOW, R1                 ; 0686
                                                        50  61 D0 00202       MOVL    (R1), TEMP_LINK                     ;
                                                            20 11 00205       BRB     23$                                 ;
                                                08  AE  50    8F 9A 00207 21$:  MOVZBL  #80, WINDOW_SIZE                  ; 0689
                                                            01 DD 0020C       PUSHL   #1                                  ; 0690
                                        51      0C  AE    06 C5 0020E       MULL3   #6, WINDOW_SIZE, R1                 ;
                                                        30  A1 9F 00213       PUSHAB  48(R1)                              ;
                                    0000G   CF            02 FB 00216       CALLS   #2, ALLOCATE                        ;
                                                        50  D5 0021B         TSTL    TEMP_LINK                           ; 0691
                                                            08 12 0021D       BNEQ    23$                                 ;
                                    0000G   CF            04 88 0021F       BISB2   #4, CLEANUP_FLAGS+1                 ; 0694
                                                    00AD    31 00224 22$:     BRW     25$                                 ; 0693
                                            12              08 DA 00227 23$:  MTPR    #8, #18                             ; 0698
                                        51      04  AE      0C C1 0022A       ADDL3   #12, NEW_WINDOW, R1                 ; 0703
                                                0C  A0      61 D0 0022F       MOVL    (R1), 12[TEMP_LINK]                 ;
                                        51      04  AE      10 C1 00233       ADDL3   #16, NEW_WINDOW, R1                 ; 0704
                                                10  A0      61 D0 00238       MOVL    (R1), 16[TEMP_LINK]                 ;
                                        51      04  AE      14 C1 0023C       ADDL3   #20, NEW_WINDOW, R1                 ; 0705
                                                14  A0      61 B0 00241       MOVW    (R1), 20[TEMP_LINK]                 ;
                                        51      04  AE      18 C1 00245       ADDL3   #24, NEW_WINDOW, R1                 ; 0706
                                                18  A0      61 D0 0024A       MOVL    (R1), 24[TEMP_LINK]                 ;
                                        51      04  AE      0B C1 0024E       ADDL3   #11, NEW_WINDOW, R1                 ; 0707
                                                0B  A0      61 90 00253       MOVB    (R1), 11[TEMP_LINK]                 ;
                                        51      04  AE      1C C1 00257       ADDL3   #28, NEW_WINDOW, R1                 ; 0708
                                                1C  A0      61 D0 0025C       MOVL    (R1), 28[TEMP_LINK]                 ;
                                        51      04  AE      20 C1 00260       ADDL3   #32, NEW_WINDOW, R1                 ; 0713
                                                        61  50 D0 00265       MOVL    TEMP_LINK, (R1)                     ;
                                        50      04  AE      0B C1 00268       ADDL3   #11, NEW_WINDOW, R0                 ; 0714
                                                        60  40 8F 88 0026D       BISB2   #64, (R0)                       ;
                                        50      04  AE      16 C1 00271       ADDL3   #22, NEW_WINDOW, R0                 ; 0715
                                                        60  58 B0 00276       MOVW    POINTER_COUNT, (R0)                 ;
                                        50      04  AE      2C C1 00279       ADDL3   #44, NEW_WINDOW, R0                 ; 0716
                                                        60  59 D0 0027E       MOVL    BASE_VBN, (R0)                      ;
                                        50          58      06 C5 00281       MULL3   #6, POINTER_COUNT, R0               ; 0717
```

```
                7E  04  AE          30  C1 0035          ADDL3   #48, NEW_WINDOW, -(SP)
                9E  24  AE          50  28 0028A         MOVC3   R0, WINDOW_BUFFER, @(SP)+
                50  04  AE          20  C1 0028F         ADDL3   #32, NEW_WINDOW, R0                 ;0718
                    04  AE          60  D0 00294         MOVL    (R0), NEW_WINDOW
                        59      10  AE  C0 00298         ADDL2   WINDOW_COUNT, BASE_VBN              ;0719
                50  04  AE          16  C1 0029C         ADDL3   #22, NEW_WINDOW, R0                 ;0720
                                    60  B4 002A1         CLRW    (R0)
                50  04  AE          2C  C1 002A3         ADDL3   #44, NEW_WINDOW, R0                 ;0721
                                    60      59  D0 002A8 MOVL    BASE_VBN, (R0)
                50  04  AE          0B  C1 002AB         ADDL3   #11, NEW_WINDOW, R0                 ;0722
                                    60      40  8F  88 002B0 BISB2   #64, (R0)
                                    12          00  DA 002B4 MTPR    #0, #18                         ;0723
                                    56      20  AE  9E 002B7 MOVAB   WINDOW_BUFFER, W_POINTER       ;0724
                                                58  D4 002BB CLRL    POINTER_COUNT                   ;0725
                                            10  AE  D4 002BD CLRL    WINDOW_COUNT                    ;0726
                                    66      14  AE  B0 002C0 24$:   MOVW    COUNT, (W_POINTER)      ;0733
                                    50          86  3C 002C4 MOVZWL  (W_POINTER)+, R0                ;0734
                                10  AE          50  C0 002C7 ADDL2   R0, WINDOW_COUNT
                                    86      18  AE  D0 002CB MOVL    LBN, (W_POINTER)+               ;0735
                                                58  D6 002CF INCL    POINTER_COUNT                   ;0737
                                            FE56    31 002D1 BRW     13$                             ;0640
                                            04  AE  D5 002D4 25$:   TSTL    NEW_WINDOW              ;0747
                                                3B  12 002D7 BNEQ    30$
                                            04  AC  D5 002D9 TSTL    WINDOW                          ;0750
                                                31  12 002DC BNEQ    29$
                                                01  DD 002DE PUSHL   #1                              ;0753
                                    50          58  D0 002E0 MOVL    POINTER_COUNT, R0
                                                03  12 002E3 BNEQ    26$
                                    50          01  D0 002E5 MOVL    #1, R0
                                    50          06  C4 002E8 26$:   MULL2   #6, R0                  ;0754
                                            30  A0  9F 002EB PUSHAB  48(R0)
                        0000G   CF              02  FB 002EE CALLS   #2, ALLOCATE
                            04  AE              50  D0 002F3 MOVL    R0, NEW_WINDOW
                                    6E      04  AE  D0 002F7 MOVL    NEW_WINDOW, PRIMARY_WINDOW      ;0753
                                                07  12 002FB BNEQ    28$                             ;0755
                        0000(   CF              04  88 002FD 27$:   BISB2   #4, CLEANUP_FLAGS+1     ;0758
                                                40  11 00302 BRB     32$                             ;0759
                50      04  AE          0B  C1 00304 28$:   ADDL3   #11, NEW_WINDOW, R0             ;0761
                                    60      40  8F  88 00309 BISB2   #64, (R0)
                                                05  11 0030D BRB     30$                             ;0750
                            04  AE      04  AC  D0 0030F 29$:   MOVL    WINDOW, NEW_WINDOW          ;0763
                                    12          08  DA 00314 30$:   MTPR    #8, #18                 ;0766
                50      04  AE          16  C1 00317         ADDL3   #22, NEW_WINDOW, R0             ;0767
                                    60      58  B0 0031C MOVW    POINTER_COUNT, (R0)
                50      04  AE          2C  C1 0031F         ADDL3   #44, NEW_WINDOW, R0             ;0768
                                    60      59  D0 00324 MOVL    BASE_VBN, (R0)
                                    58      06  C4 00327 MULL2   #6, R8                            ;0769
                56      04  AE          30  C1 0032A         ADDL3   #48, NEW_WINDOW, R6
                66      20  AE          58  28 0032F         MOVC3   R8, WINDOW_BUFFER, (R6)
                                    12          00  DA 00334 MTPR    #0, #18                         ;0770
                                            04  AC  D5 00337 TSTL    WINDOW                          ;0772
                                                04  12 0033A BNEQ    31$
                                    50          6E  D0 0033C MOVL    PRIMARY_WINDOW, R0              ;0774
                                                    04 0033F RET
                                    50          01  D0 00340 31$:   MOVL    #1, R0
                                                    04 00343 RET
                                                50  D4 00344 32$:   CLRL    R0                      ;0776
```

```
                                04 00346        RET                                             ;

; Routine Size:  839 bytes,    Routine Base:  $LOCKEDC1$ + 0000
```

```
;  465     0777  1 GLOBAL ROUTINE MARK_INCOMPLETE (FIRST_BLOCK) : NOVALUE =
;  466     0778  1
;  467     0779  1 !++
;  468     0780  1 !
;  469     0781  1 ! FUNCTIONAL DESCRIPTION:
;  470     0782  1 !
;  471     0783  1 !       This routine starts from the specified window and marks all
;  472     0784  1 !       successive windows in the list as being incomplete.  The list
;  473     0785  1 !       may start with the CURRENT_WINDOW; in which case the current
;  474     0786  1 !       and all successive window segments are marked as incomplete.
;  475     0787  1 !       It may also start with the PRIMARY_FCB; in which case all the
;  476     0788  1 !       window segments associated with the file are marked incomplete.
;  477     0789  1 !
;  478     0790  1 ! CALLING SEQUENCE:
;  479     0791  1 !       MARK_INCOMPLETE (ARG1)
;  480     0792  1 !
;  481     0793  1 ! INPUT PARAMETERS:
;  482     0794  1 !       ARG1: address of the first block in the list (either a WCB or FCB)
;  483     0795  1 !
;  484     0796  1 ! IMPLICIT INPUTS:
;  485     0797  1 !       none
;  486     0798  1 !
;  487     0799  1 ! OUTPUT PARAMETERS:
;  488     0800  1 !       none
;  489     0801  1 !
;  490     0802  1 ! IMPLICIT OUTPUTS:
;  491     0803  1 !       none
;  492     0804  1 !
;  493     0805  1 ! ROUTINE VALUE:
;  494     0806  1 !       none
;  495     0807  1 !
;  496     0808  1 ! SIDE EFFECTS:
;  497     0809  1 !       none
;  498     0810  1 !
;  499     0811  1 !--
;  500     0812  1
;  501     0813  2 BEGIN
;  502     0814  2
;  503     0815  2 MAP
;  504     0816  2         FIRST_BLOCK     : REF BBLOCK;           ! address of the first block
;  505     0817  2
;  506     0818  2 LOCAL
;  507     0819  2         STARTP          : REF BBLOCK,           ! address of the queue head
;  508     0820  2         POINTER         : REF BBLOCK;           ! address of the current block
;  509     0821  2
;  510     0822  2 ! Determine what type of block the first one is.  If it is an FCB then it is
;  511     0823  2 ! necessary to traverse the WCB queue and mark all the WCB's associated with
;  512     0824  2 ! the FCB as incomplete.  If it is a WCB then simply follow the links marking
;  513     0825  2 ! the WCB's as incomplete until a zero link is found.
;  514     0826  2 !
;  515     0827  2
;  516     0328  2 SELECTONE .FIRST_BLOCK[WCB$B_TYPE] OF
;  517     0829  2     SET
;  518     0830  2
;  519     0831  3     [DYN$C_WCB]:        BEGIN
;  520     0832  3                         POINTER = .FIRST_BLOCK;
;  521     0833  3                         SET_IPL (IPL$_SYNCH);
```

```
;  522     0834  3                    DO
;  523     0835  4                        BEGIN
;  524     0836  4                        IF .POINTER[WCB$B_TYPE] NEQ DYN$C_WCB
;  525     0837  4                        THEN BUG_CHECK (NOTWCBWCB, FATAL, 'Currupted WCB list');
;  526     0838  4                        POINTER[WCB$V_COMPLETE] = 0;
;  527     0839  4                        POINTER = .POINTER[WCB$L_LINK];
;  528     0840  4                        END
;  529     0841  3                    UNTIL .POINTER EQL 0;
;  530     0842  3                    SET_IPL (0);
;  531     0843  3                    RETURN;
;  532     0844  2                    END;
;  533     0845  2
;  534     0846  3        [DYN$C_FCB]:    BEGIN
;  535     0847  3                    POINTER = .FIRST_BLOCK;
;  536     0848  3                    STARTP = POINTER[FCB$L_WLFL];
;  537     0849  3                    SET_IPL (IPL$_SYNCH);
;  538     0850  3                    POINTER = .POINTER[FCB$L_WLFL];
;  539     0851  3                    UNTIL .POINTER EQLA .STARTP
;  540     0852  3                    DO
;  541     0853  4                        BEGIN
;  542     0854  4                        IF .POINTER[WCB$B_TYPE] NEQ DYN$C_WCB
;  543     0855  4                        THEN BUG_CHECK (NOTWCBWCB, FATAL, 'Currupted WCB list');
;  544     0856  4                        POINTER[WCB$V_COMPLETE] = 0;
;  545     0857  4                        POINTER = .POINTER[WCB$L_WLFL];
;  546     0858  3                        END;
;  547     0859  3                    SET_IPL (0);
;  548     0860  3                    RETURN;
;  549     0861  2                    END;
;  550     0862  2
;  551     0863  2        [OTHERWISE]:    BUG_CHECK (WCBFCBMNG, FATAL, 'WCB/FCB correspondence broken');
;  552     0864  2
;  553     0865  2        TES
;  554     0866  2
;  555     0867  1 END;                                    ! end of routine MARK_INCOMPLETE


                                        .EXTRN  BUG$_NOTWCBWCB

                        0004 00000      .ENTRY  MARK_INCOMPLETE, Save R2                ; 0777
                51      04   AC  D0 00002  MOVL    FIRST_BLOCK, R1                    ; 0828
                52      0A   A1  9A 00006  MOVZBL  10(R1), R2
                12           52  91 0000A  CMPB    R2, #18                            ; 0831
                             1C  12 0000D  BNEQ    3$
                50           51  D0 0000F  MOVL    R1, POINTER                        ; 0832
                12           08  DA 00012  MTPR    #8, #18                            ; 0833
                12      0A   A0  91 00015 1$:   CMPB    10(POINTER), #18             ; 0836
                             04  13 00019  BEQL    2$
                               FEFF 0001B  BUGW                                       ; 0837
                              0000* 0001D  .WORD   <BUG$_NOTWCBWCB!4>
            0B  A0           20  8A 0001F 2$:   BICB2   #32, T1(POINTER)            ; 0838
                50      20   A0  D0 00023  MOVL    32(POINTER), POINTER               ; 0839
                             EC  12 00027  BNEQ    1$                                 ; 0841
                             2B  11 00029  BRB     6$                                 ; 0842
                07           52  91 0002B 3$:   CMPB    R2, #7                       ; 0846
                             2A  12 0002E  BNEQ    7$
                50           51  D0 00030  MOVL    R1, POINTER                        ; 0847
```

```
                51     10  A0  9E 00033       MOVAB   16(POINTER), STARTP                   ; 0848
                12         08  DA 00037       MTPR    #8, #18                               ; 0849
                50     10  A0  D0 0003A       MOVL    16(POINTER), POINTER                  ; 0850
                51         50  D1 0003E 4$:   CMPL    POINTER, STARTP                       ; 0851
                           13  13 00041       BEQL    6$                                    ;
                12     0A  A0  91 00043       CMPB    10(POINTER), #18                      ; 0854
                           04  13 00047       BEQL    5$                                    ;
                             FEFF 00049       BUGW                                          ; 0855
                            0000* 0004B       .WORD   <BUG$_NOTWCBWCB!4>                    ;
        0B      A0         20  8A 0004D 5$:   BICB2   #32, 11(POINTER)                      ; 0856
                50         60  D0 00051       MOVL    (POINTER), POINTER                    ; 0857
                           E8  11 00054       BRB     4$                                    ; 0851
                12         00  DA 00056 6$:   MTPR    #0, #18                               ; 0859
                               04 00059       RET                                           ; 0846
                             FEFF 0005A 7$:   BUGW                                          ; 0863
                            0000* 0005C       .WORD   <BUG$_WCBFCBMNG.4>                    ;
                               04 0005E       RET                                           ; 0867

; Routine Size:  95 bytes,    Routine Base:  $LOCKEDC1$ + 0347
```

```
;  557     0868  1 GLOBAL ROUTINE MARK_COMPLETE (WINDOW) : NOVALUE =
;  558     0869  1
;  559     0870  1 !++
;  560     0871  1 !
;  561     0872  1 ! ROUTINE DESCRITION:
;  562     0873  1 !
;  563     0874  1 !       This routine is used to mark the specified window as complete.
;  564     0875  1 !       It must be execured in kernel mode.
;  565     0876  1 !
;  566     0877  1 ! CALLING SEQUENCE:
;  567     0878  1 !       MARK_COMPLETE (ARG1)
;  568     0879  1 !
;  569     0880  1 ! INPUT PARAMETERS:
;  570     0881  1 !       ARG1: address of the window list to mark
;  571     0882  1 !
;  572     0883  1 ! IMPLICIT INPUTS:
;  573     0884  1 !       none
;  574     0885  1 !
;  575     0886  1 ! OUTPUT PARAMETERS:
;  576     0887  1 !       none
;  577     0888  1 !
;  578     0889  1 ! IMPLICIT OUTPUTS:
;  579     0890  1 !       none
;  580     0891  1 !
;  581     0892  1 ! ROUTINE VALUE:
;  582     0893  1 !       none
;  583     0894  1 !
;  584     0895  1 ! SIDE EFFECTS:
;  585     0896  1 !       none
;  586     0897  1 !
;  587     0898  1 !--
;  588     0899  1
;  589     0900  2 BEGIN
;  590     0901  2
;  591     0902  2 MAP
;  592     0903  2         WINDOW          : REF BBLOCK;           ! address of the window to mark
;  593     0904  2
;  594     0905  2 LOCAL
;  595     0906  2         NEXT_SEGMENT    : REF BBLOCK;           ! address of the next segment
;  596     0907  2
;  597     0908  2 NEXT_SEGMENT = .WINDOW;
;  598     0909  2
;  599     0910  2 SET_IPL (IPL$_SYNCH);
;  600     0911  2
;  601     0912  2 DO
;  602     0913  3     BEGIN
;  603     0914  3     NEXT_SEGMENT[WCB$V_COMPLETE] = 1;           ! mark as complete
;  604     0915  3     NEXT_SEGMENT = .NEXT_SEGMENT[WCB$L_LINK];
;  605     0916  3     END
;  606     0917  2 UNTIL .NEXT_SEGMENT EQL 0;
;  607     0918  2
;  608     0919  2 SET_IPL (0);
;  609     0920  2
;  610     0921  2 RETURN;
;  611     0922  2
;  612     0923  1 END;                                            ! end of routine MARK_COMPLETE
```

```
                                   0000 00000         .ENTRY  MARK_COMPLETE, Save nothing                 ; 0868
                50        04    AC   D0 00002         MOVL    WINDOW, NEXT_SEGMENT                        ; 0908
                12              08   DA 00006         MTPR    #8, #18                                     ; 0910
          0B    A0              20   88 00009 1$:     BISB2   #32, 11(NEXT_SEGMENT)                       ; 0914
                50        20    A0   D0 0000D         MOVL    32(NEXT_SEGMENT), NEXT_SEGMENT              ; 0915
                                F6   12 00011         BNEQ    1$                                          ; 0917
                12              00   DA 00013         MTPR    #0, #18                                     ; 0919
                                     04 00016         RET                                                 ; 0923

; Routine Size:  23 bytes,    Routine Base:  $LOCKEDC1$ + 03A6


;  613          0924  1
;  614          0925  1 END
;  615          0926  0 ELUDOM




;                                       PSECT SUMMARY
;
;       Name                        Bytes                       Attributes
;
;  $LOCKEDC1$                           957  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        31         0      1000           00:01.9




;                                        COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:WITURN/OBJ=OBJ$:WITURN MSRC$:WITURN/UPDATE=(ENH$:WITURN)

; Size:          957 code + 0 data bytes
; Run Time:         00:23.6
; Elapsed Time:     01:11.1
; Lines/CPU Min:     2359
; Lexemes/CPU-Min: 12336
; Memory Used:  267 pages
```

```
_$25
Symb
----
CURR
CURR
CURR
DALL
DATA
DATA
DEAC
DEAC
DEAL
DEAL
DELE
DELE
DELE
DEL_
DEQ_
DIR_
DIR_
DIR_
DIR_
DIR_
DIR_
DISP
DISP
DUMM
ENTE
ERAS
ERL$
ERL$
ERR_
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXE$
EXTE
```

```
; Compilation Complete
```

```
_$25

Symb
----

EXTE
EXTE
EXTE
FID_
FILS
FILE
FILE
FILE
FILL
FIND

FIND
FINI
FLUS
FMGS
FREE
FULL
GET_

GET_
GET_
GET_


GET_
GET_
GET_
GET_
HEAD
IMPU
IMPU
IMPU
INIT
INIT
INIT


INIT
INIT
INIT
INVA


IOCS
IOCS
IOCS
IOCS
IOCS
IOCS
IOCS
IOCS
```

FCPDEF
B32

ACLCNTRL
LIS

ACLSUBR
LIS

SNDERL
LIS

ACPCNTRL
LIS

F11X

F11BXQP
MAP

ACCESS
LIS

TRUNC
LIS

FILESERV
MAP

WITURN
LIS

SNDSMB
LIS