```
FFFFFFFFFFFFFFF       111         111     XXX         XXX
FFFFFFFFFFFFFFF       111         111     XXX         XXX
FFFFFFFFFFFFFFF       111         111     XXX         XXX
FFF                111111      111111     XXX         XXX
FFF                111111      111111     XXX         XXX
FFF                111111      111111     XXX         XXX
FFF                   111         111        XXX   XXX
FFF                   111         111        XXX   XXX
FFF                   111         111        XXX   XXX
FFFFFFFF.FFF          111         111           XXX
FFFFFFFFFFFF          111         111           XXX
FFFFFFFFFFFF          111         111           XXX
FFF                   111         111        XXX   XXX
FFF                   111         111        XXX   XXX
FFF                   111         111        XXX   XXX
FFF                   111         111     XXX         XXX
FFF                   111         111     XXX         XXX
FFF                   111         111     XXX         XXX
FFF                111111111   111111111  XXX         XXX
FFF                111111111   111111111  XXX         XXX
FFF                111111111   111111111  XXX         XXX
```

```
_$25
Symb
----
IOC$
IO_C
IO_C
IO_D
IO_F
IO_S
KILL

KILL
KILL
LB_E
LB_D
LB_F
LB_H
LB_L
LOCA
LOCA
LOCK

LOCK
LOCK
LOCK
LOC_
LOC_
L_CO
L_CO
L_DA
L_DA
MAIN
MAKE
MAKE
MAKE
MAKE
MAKE
MAKE

MAKE
MAKE
MAP_
MAP_

MAP_
MARK
MARK
MARK
MARK
MARK
```

```
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP      AAAAAA    TTTTTTTTTT    CCCCCCCC  HH      HH
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP      AAAAAA    TTTTTTTTTT    CCCCCCCC  HH      HH
DD      DD      II      SS        PP      PP    AA    AA      TT      CC            HH      HH
DD      DD      II      SS        PP      PP    AA    AA      TT      CC            HH      HH
DD      DD      II      SS        PP      PP    AA    AA      TT      CC            HH      HH
DD      DD      II      SS        PP      PP    AA    AA      TT      CC            HH      HH
DD      DD      II        SSSSSS  PPPPPPPP    AA    AA      TT      CC            HHHHHHHHHH
DD      DD      II        SSSSSS  PPPPPPPP    AA    AA      TT      CC            HHHHHHHHHH
DD      DD      II              SS PP          AAAAAAAAAA    TT      CC            HH      HH
DD      DD      II              SS PP          AAAAAAAAAA    TT      CC            HH      HH
DD      DD      II              SS PP          AA    AA      TT      CC            HH      HH
DD      DD      II              SS PP          AA    AA      TT      CC            HH      HH
DDDDDDDD      IIIIII    SSSSSSSS  PP          AA    AA      TT        CCCCCCCC    HH      HH
DDDDDDDD      IIIIII    SSSSSSSS  PP          AA    AA      TT        CCCCCCCC    HH      HH


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
```

Table of contents

```
0000     1          .TITLE  DISPATCH - XQP dispatch routine
0000     2          .IDENT  'V04-000'
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28
0000    29  ;++
0000    30  ;
0000    31  ; FACILITY: F11B XQP
0000    32  ;
0000    33  ; ABSTRACT:
0000    34  ;       This module contains the initialization code for the XQP
0000    35  ;       and the dispatch routine which starts the thread
0000    36  ;       of execution for each XQP request.
0000    37  ;
0000    38  ;
0000    39  ; ENVIRONMENT: Kernel mode, AST level
0000    40  ;
0000    41  ;
0000    42  ;--
0000    43  ;
0000    44  ; AUTHOR: Christian Saether      , CREATION DATE: 30-July-1982
0000    45  ;
0000    46  ; MODIFIED BY:
0000    47  ;
0000    48  ;       V03-010 ACG0438         Andrew C. Goldstein,    2-Aug-1984  22:50
0000    49  ;               Don't clear caches when releasing blocking lock
0000    50  ;
0000    51  ;       V03-009 CDS0008         Christian D. Saether    15-Jul-1984
0000    52  ;               Also handle subfunction cpu and page faults which
0000    53  ;               CDS0007 neglected.
0000    54  ;
0000    55  ;       V03-008 CDS0007         Christian D. Saether    11-Apr-1984
0000    56  ;               Save and restore cpu and page fault info during
0000    57  ;               stalls.
```

```
0000     58 ;
0000     59 ;       V03-007 ACG0408         Andrew C. Goldstein,    23-Mar-1984  14:02
0000     60 ;               Make impure storage fully based
0000     61 ;
0000     62 ;       V03-006 CDS0006         Christian D. Saether    15-Dec-1983
0000     63 ;               Fix broken references to COMMON.
0000     64 ;               Load COMMON base register before calling DISPATCHER.
0000     65 ;               Play games with our CCB in the routines here.
0000     66 ;
0000     67 ;       V03-005 CDS0005         Christian D. Saether    17-Oct-1983
0000     68 ;               Last one out after requests are blocked needs
0000     69 ;               to invalidate FID and extent caches.
0000     70 ;
0000     71 ;       V03-004 CDS0004         Christian D. Saether    11-Oct-1983
0000     72 ;               Add routines to test for blocking of file system
0000     73 ;               activity.
0000     74 ;
0000     75 ;       V03-003 CDS0003         Christian D. Saether    11-Mar-1983
0000     76 ;               INSQUE the irp onto the xqp_queue here at kernel
0000     77 ;               ast level instead of in the ipl 2 exe$qxqppkt routine.
0000     78 ;               This avoids a problem where the irp is dequeued from
0000     79 ;               the xqp_queue before the ast is delivered.
0000     80 ;
0000     81 ;       V03-002 CDS0002         Christian D. Saether    31-Jan-1983
0000     82 ;               Save registers on initial call to DISPATCH.
0000     83 ;
0000     84 ;       V03-001 CDS0001         C Saether               21-Oct-1982
0000     85 ;               Add deletion deferral interlock using PCB$B_DPC.
0000     86 ;
0000     87 ;**
0000     88
0000     89                 .SBTTL  DECLARATIONS
0000     90 ;
0000     91 ; INCLUDE FILES:
0000     92 ;
0000     93                 $CCBDEF
0000     94                 $IPLDEF
0000     95                 $IRPDEF
0000     96                 $PCBDEF
0000     97                 $PHDDEF
0000     98                 $RVTDEF
0000     99                 $UCBDEF
0000    100                 $VCADEF
0000    101                 $VCBDEF
```

```
                        .SBTTL  INITXQP
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION
                ;       Call XQP one time initialization routine.  This must be the first
                ;       location in the XQP image as it is jumped to from the XQP loading
                ;       code.
                ;
                ; CALLING SEQUENCE
                ;
                ;       JMP     IN'TXQP
                ;
                ; SIDE EFFECTS
                ;
                ;       XQP is initialized.
                ;--
                        .PSECT  $CODE$,NOWRT,EXE,LONG
                INITXQP::
                        $CMKRNL_S       ROUTIN = INIT_FCP ; Do initialization.
                        RET                               ; Exit
```

```
                                0010   127          .SBTTL  DISPATCH
                                0010   128
                                0010   129 ;++
                                0010   130 ;
                                0010   131 ; FUNCTIONAL DESCRIPTION:
                                0010   132 ;       Switch to XQP kernel stack, load impure context base address,
                                0010   133 ;       call main dispatch routine, reset stack on return.
                                0010   134 ;
                                0010   135 ;
                                0010   136 ; CALLING SEQUENCE:
                                0010   137 ;       BSB SCH$QAST    ASTADR = DISPATCH       ASTPRM = IRP
                                0010   138 ;
                                0010   139 ; INPUT PARAMETERS:
                                0010   140 ;       4(AP) - ast parameter is the address of the irp to be processed.
                                0010   141 ;
                                0010   142 ; IMPLICIT INPUTS:
                                0010   143 ;       NONE
                                0010   144 ;
                                0010   145 ; OUTPUT PARAMETERS:
                                0010   146 ;       NONE
                                0010   147 ;
                                0010   148 ; IMPLICIT OUTPUTS:
                                0010   149 ;       NONE
                                0010   150 ;
                                0010   151 ; COMPLETION CODES:
                                0010   152 ;       NONE
                                0010   153 ;
                                0010   154 ; SIDE EFFECTS:
                                0010   155 ;       XQP request is processed
                                0010   156 ;
                                0010   157 ;--
                                0010   158
                                0010   159
                        0FFC    0010   160          .ENTRY  DISPATCH, ^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
5A  00000000'9F  00000000'8F    C3  0012   161          SUBL3   #XQP_QUEUE,@#CTL$GL_F11BXQP,R10 ; Set up base register
                   50    04 AC    D0  001E   162          MOVL    4(AP), R0                       ; Pick up irp address.
                 51    0004'CA    9E  0022   163          MOVAB   W^XQP_QUEUE+4(R10), R1          ; Get address of work queue.
                   00 B1    60    0E  0027   164          INSQUE  IRP$L_IOQFL(R0), @(R1)          ; Insert onto tail of queue.
                      0000'CA    D5  002B   165          TSTL    W^IO_PACKET(R10)                ; Is a request currently being
                                     002F   166                                                  ; processed?
                             01    13  002F   167          BEQL    10$                             ; EQL no, so continue.
                                04  0031   168          RET                                     ; Return.
                                     0032   169                                                  ; DISPATCHER starts this request
                                     0032   170                                                  ; when previous threads end.
           50    00000000'GF    D0  0032   171 10$:     MOVL    G^SCH$GL_CURPCB, R0             ; Get our PCB address.
                       2A A0    96  0039   172          INCB    PCB$B_DPC(R0)                   ; Disallow process deletion.
                                     003C   173
                 50    0000'CA    D0  003C   174          MOVL    W^IO_CCB(R10), R0               ; Get address of our CCB.
                   09 A0    01    90  0041   175          MOVB    #1, CCB$B_AMOD (R0)             ; Make it a kernel channel.
       0000'CA   00000000'GF    D0  0045   176          MOVL    G^CTL$AL_STACK, W^PREV_STKLIM(R10) ; Save current stack base
       0004'CA   00000000'GF    D0  004E   177          MOVL    G^CTL$AL_STACKLIM, W^PREV_STKLIM+4(R10) ; Save current stack limit
                0000'CA    5D    D0  0057   178          MOVL    FP, W^PREV_FP(R10)              ; Save current frame pointer
       00000000'GF    0000'CA    D0  005C   179          MOVL    W^XQP_STKLIM(R10), G^CTL$AL_STACK ; Set new stack base
       00000000'GF    0004'CA    D0  0065   180          MOVL    W^XQP_STKLIM+4(R10), G^CTL$AL_STACKLIM ; Set new stack limit
                 5E    0000'CA    D0  006E   181          MOVL    W^XQP_STKLIM(R10), SP           ; Set new stack pointer
                                     0073   182
            00000000'EF    00    FB  0073   183          CALLS   #0, DISPATCHER                  ; Call main dispatch routine
```

```
                                007A   184
                                007A   185 ;
                                007A   186 ; Note that at this point the FP was restored from tne first call
                                007A   187 ; frame on the private stack.  If this thread stalled for any reason
                                007A   188 ; that FP doesn't make any sense as it points back to a frame on the
                                007A   189 ; real stack that doesn't exist anymore.
                                007A   190 ;
                                007A   191
00000000'GF    00C0'CA   D0     007A   192          MOVL    W^PREV_STKLIM(R10), G^CTL$AL_STACK ; Restore stack base
00000000'GF    0004'CA   D0     0083   193          MOVL    W^PREV_STKLIM+4(R10), G^CTL$AL_STACKLIM ; Restore stack limit
          5D   0000'CA   D0     008C   194          MOVL    W^PREV_FP(R10), FP                 ; Restore frame pointer
                                0091   195
      50   00000000'GF   D0     0091   196          MOVL    G^SCH$GL_CURPCB, R0                ; Get our PCB address.
                 2A A0   97     0098   197          DECB    PCB$B_DPC(R0)                      ; Allow deletion again.
                                009B   198
          50   0000'CA   D0     009B   199          MOVL    W^IO_CCB(R10), R0                  ; Get our CCB address again.
            09 A0   01   8E     00A0   200          MNEGB   #1, CCB$B_AMOD (R0)                ; Make it untouchable.
                                00A4   201
                          04    00A4   202          RET                                        ; Exit.
                                00A5   203
```

```
                                00A5   205          .SBTTL  WAIT_FOR_AST - Exit current AST to await completion AST
                                00A5   206 ;++
                                00A5   207 ;
                                00A5   208 ; FUNCTIONAL DESCRIPTION:
                                00A5   209 ;       Switch back to the real kernel stack from our private stack
                                00A5   210 ;       and execute a return to dismiss the current AST.
                                00A5   211 ;       This is paired with the CONTINUE_THREAD routine which is specified
                                00A5   212 ;       as the completion AST for the service awaited.
                                00A5   213 ;
                                00A5   214 ; CALLING SEQUENCE:
                                00A5   215 ;       CALLS   #0, WAIT_FOR_AST
                                00A5   216 ;
                                00A5   217 ; IMPLICIT INPUTS:
                                00A5   218 ;       PREV_STKLIM [2] - previous kernel stack limits
                                00A5   219 ;       PREV_FP - previous kernel stack frame pointer
                                00A5   220 ;       FP - current FP on private kernel stack
                                00A5   221 ;
                                00A5   222 ; IMPLICIT OUTPUTS:
                                00A5   223 ;       XQP_SAVFP - current FP on private kernel stack
                                00A5   224 ;       CTL$AL_STACK - kernel stack base
                                00A5   225 ;       CTL$AL_STACKLIM - kernel stack limit
                                00A5   226 ;
                                00A5   227 ; NOTE: This routine expects to be currently operating on the private
                                00A5   228 ;       XQP kernel stack and is called from kernel mode.
                                00A5   229 ;
                                00A5   230 ;--
                          0BFC  00A5   231          .ENTRY  WAIT_FOR_AST,^M<R2,R3,R4,R5,R6,R7,R8,R9,R11>
              50  0000'CA   D0  00A7   232          MOVL    W^IO_CCB(R10), R0       ; Get our CCB address.
                09 A0   01   8E  00AC   233          MNEGB   #1, CCB$B_AMOD (R0)     ; Make it untouchable.
          50  00000000'GF   D0  00B0   234          MOVL    G^CTL$GL_PHD, R0        ; Get process header.
0000'CA    38 A0   0000'CA   C3  00B7   235          SUBL3   W^PMS_FNC_CPU(R10), PHD$L_CPUTIM(R0), W^PMS_FNC_CPU(R10)
                                00C0   236                                          ; Store cpu accumulated so far.
0000'CA    4C A0   0000'CA   C3  00C0   237          SUBL3   W^PMS_FNC_PFA(R10), PHD$L_PAGEFLTS(R0), W^PMS_FNC_PFA(R10)
                                00C9   238                                          ; Store pagefaults so far.
0000'CA    38 A0   0000'CA   C3  00C9   239          SUBL3   W^PMS_SUB_CPU(R10), PHD$L_CPUTIM(R0), W^PMS_SUB_CPU(R10)
                                00D2   240                                          ; Store cpu accumulated so far.
0000'CA    4C A0   0000'CA   C3  00D2   241          SUBL3   W^PMS_SUB_PFA(R10), PHD$L_PAGEFLTS(R0), W^PMS_SUB_PFA(R10)
                                00DB   242                                          ; Store pagefaults so far.
              0000'CA   5D   D0  00DB   243          MOVL    FP, W^XQP_SAVFP(R10)    ; Save current FP.
    00000000'GF   0000'CA   D0  00E0   244          MOVL    W^PREV_STKLIM(R10), G^CTL$AL_STACK ; Restore previous kernel stack b
    00000000'GF   0004'CA   D0  00E9   245          MOVL    W^PREV_STKLIM+4(R10), G^CTL$AL_STACKLIM ; Restore prev krn stack lim
              5D   0000'CA   D0  00F2   246          MOVL    W^PREV_FP(R10), FP      ; Restore previous kernel FP.
                           04  00F7   247          RET                             ; Exit from current AST.
                               00F8   248
```

```
                        00F8   250          .SBTTL  CONTINUE_THREAD - Resume thread of execution from completion AST
                        00F8   251 ;++
                        00F8   252 ;
                        00F8   253 ; FUNCTIONAL DESCRIPTION:
                        00F8   254 ;       This routine complements the above WAIT_FOR_AST routine.  It is
                        00F8   255 ;       specified as the completion AST for QIO or other services which
                        00F8   256 ;       must wait for some event.  It resets the kernel stack limits to
                        00F8   257 ;       the XQP private kernel stack and restores the saved frame pointer.
                        00F8   258 ;       It then returns to resume execution of the request at the point
                        00F8   259 ;       following the WAIT_FOR_AST call.
                        00F8   260 ;
                        00F8   261 ; CALLING SEQUENCE:
                        00F8   262 ;       ASTADR = CONTINUE_THREAD
                        00F8   263 ;
                        00F8   264 ; IMPLICIT INPUTS:
                        00F8   265 ;       CTL$AL_STACK - current kernel stack base
                        00F8   266 ;       CTL$AL_STACKLIM - current kernel stack limit
                        00F8   267 ;       XQP_STKLIM [2] - private kernel stack limits
                        00F8   268 ;       XQP_SAVFP - saved frame pointer from private kernel stack
                        00F8   269 ;       FP - current frame pointer on kernel stack
                        00F8   270 ;
                        00F8   271 ; IMPLICIT OUTPUTS:
                        00F8   272 ;       PREV_STKLIM [2] - saved limits of kernel stack on entry
                        00F8   273 ;       PREV_FP - saved kernel stack frame pointer on entry
                        00F8   274 ;       CTL$AL_STACK - set to base of private kernel stack
                        00F8   275 ;       CTL$AL_STACKLIM - set to private kernel stack limit
                        00F8   276 ;       FP - set to saved FP on private kernel stack
                        00F8   277 ;
                        00F8   278 ;--
                        00F8   279
                00000004 00F8   280 ASTPARAM = 4
                        00F8   281
                        0FFC   00F8   282          .ENTRY  CONTINUE_THREAD,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
                    5A   04 AC    D0   00FA   283          MOVL    ASTPARAM(AP),R10                        ; Set up impure area
    0000'CA   00000000'GF    D0   00FE   284          MOVL    G^CTL$AL_STACK, W^PREV_STKLIM(R10) ; Save current kernel stack base
    0004'CA   00000000'GF    D0   0107   285          MOVL    G^CTL$AL_STACKLIM, W^PREV_STKLIM+4(R10) ; Save current kern stk limi
            0000'CA   5D    D0   0110   286          MOVL    FP, W^PREV_FP(R10)              ; Save current kernel stack frame ptr
    00000000'GF   0000'CA    D0   0115   287          MOVL    W^XQP_STKLIM(R10), G^CTL$AL_STACK ; Set private kern stk base
    00000000'GF   0004'CA    D0   011E   288          MOVL    W^XQP_STKLIM+4(R10), G^CTL$AL_STACKLIM ; Set private kern stk limit
            5D   0000'CA    D0   0127   289          MOVL    W^XQP_SAVFP(R10), FP            ; Restore private kern stack frame ptr
        50   00000000'GF    D0   012C   290          MOVL    G^CTL$GL_PHD, R0                ; Get process header.
0000'CA   38 A0   0000'CA    C3   0133   291          SUBL3   W^PMS_FNC_CPU(R10), PHD$L_CPUTIM(R0), W^PMS_FNC_CPU(R10)
                        013C   292                                                  ; Restore adjusted cpu time.
0000'CA   4C A0   0000'CA    C3   013C   293          SUBL3   W^PMS_FNC_PFA(R10), PHD$L_PAGEFLTS(R0), W^PMS_FNC_PFA(R10)
                        0145   294                                                  ; Restore adjusted page faults.
0000'CA   38 A0   0000'CA    C3   0145   295          SUBL3   W^PMS_SUB_CPU(R10), PHD$L_CPUTIM(R0), W^PMS_SUB_CPU(R10)
                        014E   296                                                  ; Restore adjusted cpu time.
0000'CA   4C A0   0000'CA    C3   014E   297          SUBL3   W^PMS_SUB_PFA(R10), PHD$L_PAGEFLTS(R0), W^PMS_SUB_PFA(R10)
                        0157   298                                                  ; Restore adjusted page faults.
            50   0000'CA    D0   0157   299          MOVL    W^IO_CCB(R10), R0               ; Get our CCB address.
              09 A0   01    90   015C   300          MOVB    #1, CCB$B_AMOD (R0)             ; Make it a useable kernel channel.
                        04   0160   301          RET                                     ; Return to stalled thread
```

```
                                0161   303          .SBTTL  START_REQUEST - test for blocking and raise activity count
                                0161   304 ;++
                                0161   305 ;--
                                0161   306 START_REQUEST::
        50    0000'CA    D0     0161   307          MOVL    W^CURRENT_VCB(R10), R0  ; get vcb addr
                                0166   308          SETIPL  20$                     ; raise ipl to test activity
                OE A0   B5      016D   309          TSTW    VCB$W_RVN (R0)          ; is this volume set?
                   14   12      0170   310          BNEQ    10$                     ; NEQ it is a volume set.
                008C C0   D5    0172   311          TSTL    VCB$L_BLOCKID (R0)      ; is block lock already armed?
                   21   13      0176   312          BEQL    30$                     ; EQL it isn't
              1C 00A0 C0   E9   0178   313          BLBC    VCB$W_ACTIVITY (R0), 30$
            00A0 C0    02   A0  017D   314          ADDW2   #2, VCB$W_ACTIVITY (R0)
                            0182   315 5$:      SETIPL  #0
                         05 0185   316          RSB
                            0186   317
                            0186   318 ;
                            0186   319 ; volume set
                            0186   320 ;
                            0186   321
        50    20 A0    D0   0186   322 10$:     MOVL    VCB$L_RVT (R0), R0
                24 A0    D5  018A   323          TSTL    RVT$L_BLOCKID (R0)
                   0A    13  018D   324          BEQL    30$
              06 06 A0    E9 018F   325          BLBC    RVT$W_ACTIVITY (R0), 30$
            06 A0    02   A0  0193   326          ADDW2   #2, RVT$W_ACTIVITY (R0)
                   E9    11  0197   327          BRB     5$
                            0199   328
                            0199   329 ;
                            0199   330 ; wait for blocking lock to be granted and rearm if it isn't already.
                            0199   331 ;
                            0199   332
                            0199   333 30$:     SETIPL  #0
00000000'EF         00      FB  019C   334          CALLS   #0, BLOCK_WAIT
                    BC      11  01A3   335          BRB     START_REQUEST
                            01A5   336
                00000008    01A5   337 20$:     .LONG   IPL$_SYNCH              ; can't page at synch
                            01A9   338
```

```
                          01A9   340          .SBTTL  FINISH_REQUEST - lower activity count, check activity allowed
                          01A9   341 ;++
                          01A9   342 ;--
                          01A9   343
                          01A9   344 FINISH_REQUEST::
51    0000'CA    D0       01A9   345          MOVL    W^CURRENT_VCB(R10), R1
                          01AE   346          SETIPL  #0$
      0E A1      B5       01B5   347          TSTW    VCB$W_RVN (R1)
         0E      12       01B8   348          BNEQ    20$
00A0 C1  02      A2       01BA   349          SUBW2   #2, VCB$W_ACTIVITY (R1)
  12 00A0 C1     E9       01BF   350          BLBC    VCB$W_ACTIVITY (R1), 40$
                          01C4   351 10$:     SETIPL  #0
                    05    01C7   352          RSB
                          01C8   353
                          01C8   354 ;
                          01C8   355 ; volume set
                          01C8   356 ;
                          01C8   357
  50    20 A1      D0       01C8   358 20$:     MOVL    VCB$L_RVT (R1), R0
  06 A0    02      A2       01CC   359          SUBW2   #2, RVT$W_ACTIVITY (R0)
    F0 06 A0      E8       01D0   360          BLBS    RVT$W_ACTIVITY (R0), 10$
          17      13       01D4   361          BEQL    110$
                          01D6   362
          EC      12       01D6   363 40$:     BNEQ    10$
                          01D8   364          SETIPL  #0
50    008C C1     9E       01DB   365          MOVAB   VCB$L_BLOCKID (R1), R0
                          01E0   366
                          01E0   367 ;
                          01E0   368 ; Only one process can see the count go to zero.
                          01E0   369 ; R0 = address of either vcb or rvt blockid field.
                          01E0   370 ;
                          01E0   371
      51    60      D0       01E0   372 100$:    MOVL    (R0), R1
            60      D4       01E3   373          CLRL    (R0)
            51      DD       01E5   374          PUSHL   R1
0000'CF     01      FB       01E7   375          CALLS   #1,W^DEQ_LOCK
                    05    01EC   376          RSB
                          01ED   377
                          01ED   378 ;
                          01ED   379 ; This is a volume set.
                          01ED   380 ; R0 = RVT.
                          01ED   381 ;
                          01ED   382
                          01ED   383 110$:    SETIPL  #0
  50    24 A0      9E       01F0   384          MOVAB   RVT$L_BLOCKID (R0), R0
           EA      11       01F4   385          BRB     100$
                          01F6   386
         00000008         01F6   387 50$:     .LONG   IPL$_SYNCH
                          01FA   388
                          01FA   389
                          01FA   390          .END
```

```
ASTPARAM                        = 00000004
BLOCK_WAIT                        ********    X   02
CCB$B_AMOD                      = 00000009
CONTINUE_THREAD                   000000F8 RG     02
CTL$AL_STACK                      ********    X   02
CTL$AL_STACKLIM                   ********    X   02
CTL$GL_F11BXQP                    ********    X   02
CTL$GL_PHD                        ********    X   02
CURRENT_VCB                       ********    X   02
DEQ_LOCK                          ********    X   02
DISPATCH                          00000010 RG     02
DISPATCHER                        ********    X   02
FINISH_REQUEST                    000001A9 RG     02
INITXQP                           00000000 RG     02
INIT_FCP                          ********    X   02
IO_CCB                            ********    X   02
IO_PACKET                         ********    X   02
IPL$_SYNCH                      = 00000008
IRP$L_IOQFL                     = 00000000
PCB$B_DPC                       = 0000002A
PHD$L_CPUTIM                    = 00000038
PHD$L_PAGEFLTS                  = 0000004C
PMS_FNC_CPU                       ********    X   02
PMS_FNC_PFA                       ********    X   02
PMS_SUB_CPU                       ********    X   02
PMS_SUB_PFA                       ********    X   02
PR$_IPL                           ********    X   02
PREV_FP                           ********    X   02
PREV_STKLIM                       ********    X   02
RVT$L_BLOCKID                   = 00000024
RVT$W_ACTIVITY                  = 00000006
SCH$GL_CURPCB                     ********    X   02
START_REQUEST                     00000161 RG     02
SYS$CMKRNL                        ********   GX   02
VCB$L_BLOCKID                   = 0000008C
VCB$L_RVT                       = 00000020
VCB$W_ACTIVITY                  = 000000A0
VCB$W_RVN                       = 0000000E
WAIT_FOR_AST                      000000A5 RG     02
XQP_QUEUE                         ********    X   02
XQP_SAVFP                         ********    X   02
XQP_STKLIM                        ********    X   02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
$CODE$	000001FA (506.)	02 (2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.08	00:00:00.54
Command processing	128	00:00:00.50	00:00:03.71
Pass 1	309	00:00:09.75	00:00:23.02
Symbol table sort	0	00:00:01.67	00:00:03.47
Pass 2	82	00:00:01.99	00:00:04.62
Symbol table output	7	00:00:00.07	00:00:00.08
Psect synopsis output	1	00:00:00.02	00:00:00.18
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	562	00:00:14.08	00:00:35.62

The working set limit was 1350 pages.
55800 bytes (109 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1072 non-local and 11 local symbols.
390 source lines were read in Pass 1, producing 23 object records in Pass 2.
19 pages of virtual memory were used to define 18 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[SYS.OBJ]LIB.MLB;1	10
_255DUA28:[SYSLIB]STARLET.MLB;2	5
TOTALS (all libraries)	15

1155 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

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

DEACCS
LIS

DELETE
LIS

DIRSCN
LIS

DISPAT
LIS

CREHDR
LIS

DIRACC
LIS

DELBAD
LIS

CREFCB
LIS

CREWIN
LIS

DISPATCH
LIS

ENTER
LIS

DELFIL
LIS