```
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
   SSSSSSSSSSSS  YYY         YYY     SSSSSSSSSSSS  LLL                 OOOOOOOOO        AAAAAAAAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS              YYY         YYY  SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
SSS                 YYY   YYY     SSS              LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
   SSSSSSSSS           YYY           SSSSSSSSS     LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAAAAAAAAAAAAAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
            SSS        YYY                    SSS  LLL              OOO         OOO  AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
SSSSSSSSSSSS           YYY        SSSSSSSSSSSS     LLLLLLLLLLLLLLL     OOOOOOOOO     AAA         AAA
```

FILEID**FALMAN


```
FFFFFFFFFF    AAAAAA    LL          MM      MM    AAAAAA    NN      NN
FFFFFFFFFF    AAAAAA    LL          MM      MM    AAAAAA    NN      NN
FF          AA      AA  LL          MMMM  MMMM  AA      AA  NN      NN
FF          AA      AA  LL          MMMM  MMMM  AA      AA  NN      NN
FF          AA      AA  LL          MM  MM  MM  AA      AA  NNNN    NN
FF          AA      AA  LL          MM  MM  MM  AA      AA  NNNN    NN
FFFFFFFF    AA      AA  LL          MM      MM  AA      AA  NN  NN  NN
FFFFFFFF    AA      AA  LL          MM      MM  AA      AA  NN  NN  NN
FF          AAAAAAAAAA  LL          MM      MM  AAAAAAAAAA  NN    NNNN
FF          AAAAAAAAAA  LL          MM      MM  AAAAAAAAAA  NN    NNNN
FF          AA      AA  LL          MM      MM  AA      AA  NN      NN      ....
FF          AA      AA  LL          MM      MM  AA      AA  NN      NN      ....
FF          AA      AA  LLLLLLLLLL  MM      MM  AA      AA  NN      NN      ....
FF          AA      AA  LLLLLLLLLL  MM      MM  AA      AA  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
```

Table of contents

```
0000          .TITLE  FALMAN - Cluster Failover Manager
0000          .IDENT  'V04-000'
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY: EXECUTIVE, CLUSTER MANAGEMENT
0000    ;
0000    ; ABSTRACT:
0000    ;       This module contains the routines that direct failover in a
0000    ;       VAXcluster.
0000    ;
0000    ; ENVIRONMENT: VAX/VMS
0000    ;
0000    ; AUTHOR: David W. Thiel,        CREATION DATE: 24-May-1983
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-014 DWT0225         David W. Thiel          11-Jul-1984
0000    ;               Change call to temporary name EXE$MNTVERSP2 to
0000    ;               real name EXE$CLUTRANIO.
0000    ;
0000    ;       V03-013 DWT0222         David W. Thiel          25-Jun-1984
0000    ;               Revise CNX$FAILO_SYNC to minimize time delays
0000    ;               in synchronizing failover steps.
0000    ;
0000    ;       V03-012 DWT0220         David W. Thiel          8-May-1984
0000    ;               Correct previous fix to handle reference to symbol
0000    ;               outside of this image.
0000    ;
0000    ;       V03-011 DWT0219         David W. Thiel          8-May-1984
0000    ;               Add synchronization between I/O and lock manager
0000    ;               clusters.
0000    ;
0000    ;       V03-010 DWT0188         David W. Thiel          9-Mar-1984
```

```
0000  ;               Correct synchronization routine to correctly drop
0000  ;               thread when it is discovered that another failover
0000  ;               has begun.
0000  ;
0000  ;       V03-009 SRB0110         Steve Beckhardt         2-Feb-1984
0000  ;               Added more failover table entries for improved
0000  ;               consistency checking.
0000  ;
0000  ;       V03-008 DWT0154         David W. Thiel          29-Dec-1983
0000  ;               Use single table for addition and removal of nodes.
0000  ;               Change names of CNX$BEGIN_FAILIN and
0000  ;               CNX$BEGIN_FAILOVER to CNX$MEMBERSHIP_CHANGE.
0000  ;               Remove obsolete failover table entries and
0000  ;               CNX$FAILO_QWAIT routine.
0000  ;
0000  ;       V03-007 DWT0137         David W. Thiel          07-Oct-1983
0000  ;               Differentiate cases of a certain future failover
0000  ;               (based on existence of a broken connection to a
0000  ;               cluster member) and a pending failover (node has
0000  ;               been failed out, but failover table processing
0000  ;               has not begun due to the state of a previous
0000  ;               instance of failover table processing) using the
0000  ;               CLUB$V_LOST_CNX bit.
0000  ;
0000  ;       V03-006 DWT0121         David W. Thiel          20-Aug-1983
0000  ;               Add failover table entry to start journal recovery.
0000  ;
0000  ;       V03-005 DWT0116         David W. Thiel          1-Aug-1983
0000  ;               Correct synchronization logic.  Correct quorum
0000  ;               waiting after a failover.
0000  ;
0000  ;       V03-004 DWT0111         David W. Thiel          27-July-1983
0000  ;               Add fail-in table.
0000  ;               Convert interface to failover routines from JMP to
0000  ;               JSB.
0000  ;
0000  ;       V03-003 SRB0095         Steve Beckhardt         9-Jun-1983
0000  ;               Add entries to failover table.
0000  ;
0000  ;       V03-002 DWT0104         David W. Thiel          8-June-1983
0000  ;               Add failover routine CNX$FAILO_QWAIT that waits for
0000  ;               quorum before completing.
0000  ;               Add CNX$CHECK_FAILOVER to test for and initiate
0000  ;               a pending failover.
0000  ;
0000  ;       V03-001 DWT0103         David W. Thiel          27-May-1983
0000  ;               Add index definition argument to FSTEP macro.
0000  ;               Add jacket routines for calling existing failover
0000  ;               routines.
0000  ;               Avoid trying to send message to a local node.
0000  ;               Fix logic error in synchronization code.
0000  ;
0000  ;--
0000
```

```
0000            .SBTTL  DECLARATIONS
0000    ;
0000    ; INCLUDE FILES:
0000    ;
0000            $CDRPDEF                                ; CDRP offsets
0000            $CLSMSGDEF                              ; Cluster message definitions
0000            $CLUBDEF                                ; CLUster Block offsets
0000            $CSBDEF                                 ; CSB Offsets
0000            $FKBDEF                                 ; Fork block offsets
0000            $IPLDEF                                 ; IPL definitions
0000            $SSDEF                                  ; Status code definitions
0000
0000    ;
0000    ; MACRO DEFINITIONS:
0000    ;
0000
0000            .MACRO  FSTEP   ADDR,STEP_INDEX,?LABEL
0000            .SHOW   BINARY
0000                    ; Self relative address
0000                    LABEL:  .LONG   ADDR-LABEL
0000            .IF NB  STEP_INDEX
0000            .SHOW   EXPANSIONS
0000                    STEP_INDEX=     <LABEL-CNX$FAILOVER_TABLE>@<-2>
0000            .NOSHOW EXPANSIONS
0000            .ENDC
0000            .NOSHOW BINARY
0000            .ENDM   FSTEP
0000
0000    ;***************** T E M P O R A R Y   D E F I N I T I O N S : *********
0000    ;
0000    .IIF NOT_DEFINED CLUFCB$V_WAITING,      CLUFCB$V_WAITING=       CLUFCB$V_FKB_BUSY+1
0000    .IIF NOT_DEFINED CLUFCB$M_WAITING,      CLUFCB$M_WAITING=       1@CLUFCB$V_WAITING
0000    ;
0000    ;***************************************************************************
0000
0000
0000    ;***************************************************************************
0000    ;
0000    ; NOTE: The following assumptions are in effect for this entire module.
0000    ;
0000    ;***************************************************************************
0000
0000            ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
0000            ASSUME  IPL$_SYNCH  EQ  IPL$_TIMER
```

```
0000          .SBTTL  CNX$FAILOVER_TABLE - Failover Sequencing Table
0000
0000  ;++
0000  ;
0000  ; FUNCTIONAL DESCRIPTION:
0000  ;
0000  ;       The failover table lists the sequence of steps in failing nodes
0000  ;       out of a cluster.  Each table entry is the self-relative address
0000  ;       of a routine that performs one step.
0000  ;
0000  ;       If any failover step breaks the thread of IPL SYNC execution, it
0000  ;       is possible that another failover may be needed at that point.
0000  ;       The way in which this is handled depends on a mode of failover
0000  ;       table processing.  In NOREFAIL mode, handling of a subsequent
0000  ;       failover until either this mode is left or the end of table
0000  ;       processing is reached.  In REFAIL mode, this failover is
0000  ;       abandoned and processing of the table entries is reinitiated
0000  ;       from the beginning.
0000  ;
0000  ;       The table is processed as follows.  Each node independently
0000  ;       executes the steps in the failover table, synchronized only at
0000  ;       the begining and when specific requests for synchronization
0000  ;       are made.  Control of synchronization and other aspects of
0000  ;       failover processing are themselves controlled by table entries.
0000  ;       These special tables entries are described next.
0000  ;
0000  ;    CNX$FAILO_NOP -- No Operation
0000  ;
0000  ;       NOP entry.  Used to reserve space for patching.
0000  ;
0000  ;    CNX$FAILO_END -- End of Table
0000  ;
0000  ;       Marks the end of the table and terminates table processing.
0000  ;
0000  ;    CNX$FAILO_REFAIL -- Set Re-failoverable Mode
0000  ;
0000  ;       Specifies that this failover may be abandoned at any time and
0000  ;       another begun.  In this mode, any suspended thread must save
0000  ;       the failover identification and validate it upon being
0000  ;       resumed.  If the ID has changed, the thread must be terminated;
0000  ;       otherwise, it may proceed.
0000  ;
0000  ;    CNX$FAILO_NOREFAIL -- Clear Re-failoverable Mode
0000  ;
0000  ;       Specifies that this failover may not be abandoned and that
0000  ;       no new failover may be begun.  While in this mode, failover
0000  ;       routines MAY NOT suspend themselves without a guarantee of
0000  ;       resumption.  In particular, a routine MAY NOT wait for a
0000  ;       message to arrive from the corresponding routine on another
0000  ;       node.  Furthermore, a routine MAY NOT wait for process
0000  ;       execution.
0000  ;
0000  ;    CNX$FAILO_SYNC -- Synchronize Failover Steps
0000  ;
0000  ;       Requests synchronization at this step.  One node is selected
0000  ;       as the synchronizer when failover table processing is
0000  ;       initiated.  When this table entry is encountered, every
```

```
0000    ;          node sends a READY message to the synchronizing node.  When
0000    ;          the synchronizing node executes this entry, it waits for a
0000    ;          READY from every other node involved in the failover and
0000    ;          then sends a DOSTEP message to every involved node.  It
0000    ;          and the other nodes then complete this step and proceed to
0000    ;          the next.
0000    ;
0000    ;          Following a synchronization step, it is guaranteed that every
0000    ;          other involved node has completed the step preceding the
0000    ;          synchronization step.  Due to delays and queuing of messages,
0000    ;          it is still possible to receive a message from another node
0000    ;          that was queued before the synchronization step.
0000    ;
0000    ;          Requesting a synchronization step ALWAYS puts the failover
0000    ;          into Re-Failoverable Mode.  This results in the very substantial
0000    ;          simplification which allows any communications problems during
0000    ;          synchronization to be handled as part of a subsequent failover.
0000    ;
0000    ; The following description applies to every failover routine:
0000    ;
0000    ; CALLING SEQUENCE:
0000    ;
0000    ;          Invoked by:
0000    ;                  JSB failover_routine
0000    ;          IPL is IPL$_SYNC = IPL$_SCS = IPL$_TIMER
0000    ;          To continue failover processing, return with:
0000    ;                  RSB
0000    ;          To abandon failover processing in REFAIL mode:
0000    ;                  ADDL    #4,SP
0000    ;                  RSB
0000    ;          To terminate failover processing in either mode:
0000    ;                  ADDL    #4,SP
0000    ;                  JMP     CNX$END_FAILOVER
0000    ;
0000    ;          A failover routine runs as a fork process and must
0000    ;          behave according to the general rules for fork processes.
0000    ;
0000    ; INPUT PARAMETERS:
0000    ;
0000    ;          04(SP): Address of caller's caller -- return here if thread
0000    ;                          is suspended
0000    ;          00(SP): Return here to continue failover processing
0000    ;
0000    ;          R5:     Address of CLUster Failover Control Block (CLUFCB)
0000    ;          R4:     Address of CLUster Block (CLUB)
0000    ;          R3:     Failover ID (copy of CLUFCB$L_ID(R5))
0000    ;
0000    ; OUTPUT PARAMETERS:
0000    ;
0000    ;          NONE
0000    ;
0000    ; COMPLETION CODES:
0000    ;
0000    ;          NONE
0000    ;
0000    ; SIDE EFFECTS:
0000    ;
```

```
          0000   ;       Registers R0-R5 need not be preserved.  Everything interesting
          0000   ;       that a failover does is a side effect of some sort.
          0000   ;
          0000   ;--
          0000
      00000000          .PSECT  $$$060,LONG                     ; Read only Data PSECT
          0000
          0000   CNX$FAILOVER_TABLE::
          0000          FSTEP   CNX$FAILO_NOREFAIL              ; Must be first entry
000000A4' 0000                  30000$: .LONG   CNX$FAILO_NOREFAIL-30000$
          0004
          0004          FSTEP   LCK$STALL_ALL                   ; Stall lock requests
FFFFFFFC' 0004                  30001$: .LONG   LCK$STALL_ALL-30001$
          0008   ;
          0008   ; Everything up to this point must be synchronous with the beginning
          0008   ; of failover table execution.  Note that the beginning of failover
          0008   ; table execution is not necessarily synchronous with the receipt of
          0008   ; the phase 2 message in the case when a failover is already in
          0008   ; progress.
          0008   ;
          0008          FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000097' 0008                  30002$: .LONG   CNX$FAILO_REFAIL-30002$
          000C          FSTEP   CNX$FAILO_SYNC                  ; Synchronize
000000B5' 000C                  30003$: .LONG   CNX$FAILO_SYNC-30003$
          0010          FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000094' 0010                  30004$: .LONG   CNX$FAILO_NOREFAIL-30004$
          0014
          0014          FSTEP   CNX$IO_SYNCH                    ; Serialize I/O
000000A7' 0014                  30005$: .LONG   CNX$IO_SYNCH-30005$
          0018
          0018          FSTEP   LCK$SET_STATE1                  ; Set rebuild state 1
FFFFFFE8' 0018                  30006$: .LONG   LCK$SET_STATE1-30006$
          001C
          001C          FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000083' 001C                  30007$: .LONG   CNX$FAILO_REFAIL-30007$
          0020          FSTEP   CNX$FAILO_SYNC                  ; Synchronize
000000A1' 0020                  30008$: .LONG   CNX$FAILO_SYNC-30008$
          0024          FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000080' 0024                  30009$: .LONG   CNX$FAILO_NOREFAIL-30009$
          0028
          0028          FSTEP   LCK$INIT_REBUILD                ; Remove all master copy locks
FFFFFFD8' 0028                  30010$: .LONG   LCK$INIT_REBUILD-30010$
          002C                                                  ;  and directory entries
          002C
          002C          FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000073' 002C                  30011$: .LONG   CNX$FAILO_REFAIL-30011$
          0030          FSTEP   CNX$FAILO_SYNC                  ; Synchronize
00000091' 0030                  30012$: .LONG   CNX$FAILO_SYNC-30012$
          0034          FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000070' 0034                  30013$: .LONG   CNX$FAILO_NOREFAIL-30013$
          0038
          0038          FSTEP   LCK$SET_STATE2                  ; Set rebuild state 2
FFFFFFC8' 0038                  30014$: .LONG   LCK$SET_STATE2-30014$
          003C
          003C          FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000063' 003C                  30015$: .LONG   CNX$FAILO_REFAIL-30015$
          0040          FSTEP   CNX$FAILO_SYNC                  ; Synchronize
```

```
00000081' 0040          30016$: .LONG   CNX$FAILO_SYNC-30016$
          0044  FSTEP   CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000060' 0044          30017$: .LONG   CNX$FAILO_NOREFAIL-30017$
          0048
          0048  FSTEP   LCK$REBUILD_LKBS        ; Rebuild locks
FFFFFFB8' 0048          30018$: .LONG   LCK$REBUILD_LKBS-30018$
          004C
          004C  FSTEP   CNX$FAILO_REFAIL        ; Must precede synchronize
00000053' 004C          30019$: .LONG   CNX$FAILO_REFAIL-30019$
          0050  FSTEP   CNX$FAILO_SYNC          ; Synchronize
00000071' 0050          30020$: .LONG   CNX$FAILO_SYNC-30020$
          0054  FSTEP   CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000050' 0054          30021$: .LONG   CNX$FAILO_NOREFAIL-30021$
          0058
          0058  FSTEP   LCK$SET_STATE3          ; Set rebuild state 3
FFFFFFA8' 0058          30022$: .LONG   LCK$SET_STATE3-30022$
          005C
          005C  FSTEP   CNX$FAILO_REFAIL        ; Must precede synchronize
00000043' 005C          30023$: .LONG   CNX$FAILO_REFAIL-30023$
          0060  FSTEP   CNX$FAILO_SYNC          ; Synchronize
00000061' 0060          30024$: .LONG   CNX$FAILO_SYNC-30024$
          0064  FSTEP   CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000040' 0064          30025$: .LONG   CNX$FAILO_NOREFAIL-30025$
          0068
          0068  FSTEP   LCK$REBUILD_RSBS        ; Rebuild resources, grant
FFFFFF98' 0068          30026$: .LONG   LCK$REBUILD_RSBS-30026$
          006C                                  ; unprotected locks
          006C
          006C  FSTEP   CNX$FAILO_REFAIL        ; Must precede synchronize
00000033' 006C          30027$: .LONG   CNX$FAILO_REFAIL-30027$
          0070  FSTEP   CNX$FAILO_SYNC          ; Synchronize
00000051' 0070          30028$: .LONG   CNX$FAILO_SYNC-30028$
          0074  FSTEP   CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000030' 0074          30029$: .LONG   CNX$FAILO_NOREFAIL-30029$
          0078
          0078  FSTEP   LCK$SET_STATE0          ; Set rebuild state 0
FFFFFF88' 0078          30030$: .LONG   LCK$SET_STATE0-30030$
          007C
          007C  FSTEP   CNX$FAILO_REFAIL        ; Must precede synchronize
00000023' 007C          30031$: .LONG   CNX$FAILO_REFAIL-30031$
          0080  FSTEP   CNX$FAILO_SYNC          ; Synchronize
00000041' 0080          30032$: .LONG   CNX$FAILO_SYNC-30032$
          0084
          0084  FSTEP   LCK$RESUME_UNPROT       ; Resume processes waiting for
FFFFFF7C' 0084          30033$: .LONG   LCK$RESUME_UNPROT-30033$
          0088                                  ; unprotected locks
          0088
          0088  FSTEP   CNX$FAILO_JNL           ; CSP/RCP recovery initiation
00000223' 0088          30034$: .LONG   CNX$FAILO_JNL-30034$
          008C
          008C  FSTEP   CNX$FAILO_SYNC          ; Synchronize
00000035' 008C          30035$: .LONG   CNX$FAILO_SYNC-30035$
          0090
          0090  FSTEP   LCK$RESUME_ALL          ; Resume processes waiting for any
FFFFFF70' 0090          30036$: .LONG   LCK$RESUME_ALL-30036$
          0094                                  ;  type of lock request
          0094
```

```
           0094      FSTEP   CNX$FAILO_NOREFAIL       ; Don't allow re-failovers
00000010'  0094              30037$: .LONG   CNX$FAILO_NOREFAIL-30037$
           0098
           0098      FSTEP   LCK$REBUILD_RSBS         ; Rebuild resources, grant
FFFFFF68'  0098              30038$: .LONG   LCK$REBUILD_RSBS-30038$
           009C                                       ;  protected (all) locks
           009C
           009C      FSTEP   CNX$FAILO_REFAIL,FORCE_END       ; Ending point
00000003'  009C              30039$: .LONG   CNX$FAILO_REFAIL-30039$
           00A0      .SHOW   EXPANSIONS
00000027   00A0              FORCE_END=      <30039$-CNX$FAILOVER_TABLE>@<-2>
           00A0      FSTEP   CNX$FAILO_END            ; End of table
00000017'  00A0              30040$: .LONG   CNX$FAILO_END-30040$
           00A4  ;
000000C4   00A4      .BLKL   8                        ; Reserve patch space
           00C4  ;
           00C4  ;
    00000000         .PSECT  $$$100,LONG
           0000
           0000      .DEFAULT        DISPLACEMENT,WORD
```

```
                              0000          .SBTTL  CNX$MEMBERSHIP_CHANGE - Begin membership change actions
                              0000  ;++
                              0000  ;
                              0000  ; FUNCTIONAL DESCRIPTION:
                              0000  ;
                              0000  ;  CNX$MEMBERSHIP_CHANGE
                              0000  ;       This routine is called whenever a node or nodes are added to
                              0000  ;       or removed from a cluster, including initial cluster formation.
                              0000  ;       Actions are undertaking by the members of the cluster to
                              0000  ;       adjust to the new cluster membership.  If an uninterruptible
                              0000  ;       action is already in progress, handling of the new event is deferred.
                              0000  ;
                              0000  ; CALLING SEQUENCE:
                              0000  ;
                              0000  ;       JSB     CNX$MEMBERSHIP_CHANGE
                              0000  ;       IPL is IPL$_SCS
                              0000  ;
                              0000  ; INPUT PARAMETERS:
                              0000  ;
                              0000  ;       NONE
                              0000  ;
                              0000  ; OUTPUT PARAMETERS:
                              0000  ;
                              0000  ;       NONE
                              0000  ;
                              0000  ; COMPLETION CODES:
                              0000  ;
                              0000  ;       NONE
                              0000  ;
                              0000  ; SIDE EFFECTS:
                              0000  ;
                              0000  ;       R0-R5 are destroyed.
                              0000  ;
                              0000  ;--
                              0000
                              0000          .ENABLE LSB
                              0000  CNX$MEMBERSHIP_CHANGE::
54   00000000'GF      D0      0000  1$:     MOVL    G^CLU$GL_CLUB,R4              ; Address of CLUB
   55   010C C4       9E      0007          MOVAB   CLUB$B_CLUFCB(R4),R5          ; Address of failover control block
 05 20 A5     00      E1      000C          BBC     #CLUFCB$V_ACTIVE, -           ; Branch if no active failover and
                              0011                  CLUFCB$L_STATUS(R5),10$ ;        set active failover flag
    20 A5     02      C8      0011          BISL2   #CLUFCB$M_PENDING, -          ; Set failover pending flag
                              0015                  CLUFCB$L_STATUS(R5)
              05              0015          RSB                                   ; Return, will do failover later
                              0016
    20 A5     06      CA      0016  10$:    BICL2   #<CLUFCB$M_PENDING ! -        ; Clear failover pending flag
                              001A                    CLUFCB$M_SYNC_NODE>, - ;       and local synchronizing node flags
                              001A                  CLUFCB$L_STATUS(R5)
  1C A5     34 A4     D0      001A          MOVL    CLUB$L_LST_XTN(R4), -         ; Make last transition ID the failover
                              001F                  CLUFCB$L_ID(R5)               ;    identifier
          18 A5       D4      001F          CLRL    CLUFCB$L_STEP(R5)             ; Initialize failover step
  24 A5     5C A4     D0      0022          MOVL    CLUB$L_COORD(R4), -           ; Set up CSB address of synchronizer node
                              0027                  CLUFCB$L_SYNC_CSB(R5)
  10 A4     5C A4     D1      0027          CMPL    CLUB$L_COORD(R4), -           ; Was this node the failover coordinator?
                              002C                  CLUB$L_LOCAL_CSB(R4)
          04          12      002C          BNEQ    20$                           ; Branch if no
    20 A5     04      C8      002E          BISL2   #CLUFCB$M_SYNC_NODE, -        ; Make this node the failover sync node
```

```
                                 0032         CLUFCB$L_STATUS(R5)
                   55    DD      0032  20$:   PUSHL   R5                          ; Save CLUFCB address
                                 0034         ASSUME  <CLUFCB$B_NODEMAP+CLUFCB$S_NODEMAP> EQ CLUFCB$B_RESPMAP
28 A5   0040 8F   00  6E   00    2C   0034    MOVC5   #0,(SP),#0, -              ; Clear node map and response map
                                 003D                 #CLUFCB$S_NODEMAP+CLUFCB$S_RESPMAP, -
                                 003D                 CLUFCB$B_NODEMAP(R5)
                   20    BA      003D         POPR    #^M<R5>                     ; Restore CLUFCB address
                FFBE'    30      003F         BSBW    CNX$SCAN_CSBS               ; Iterate over all CSBs
                24 50    E9      0042         BLBC    R0,40$                      ; Branch when done
         0E 60 A3  01    E1      0045         BBC     #CSB$V_MEMBER, -            ; Branch if CSB is not for a member node
                                 004A                 CSB$L_STATUS(R3),30$
         09 60 A3  00    E0      004A         BBS     #CSB$V_LONG_BREAK, -        ; Branch if connection permanently broken
                                 004F                 CSB$L_STATUS(R3),30$
            50  4C A3  3C        004F         MOVZWL  CSB$W_CSID_IDX(R3),R0       ; CSID index of member node
         00 28 A5  50    E3      0053         BBCS    R0, -                       ; Mark node present
                                 0058                 CLUFCB$B_NODEMAP(R5),30$
                   05            0058  30$:   RSB
                                 0059
                                 0059  ;
                                 0059  ; Failover routine may jump here to exit failover processing
                                 0059  ;
                                 0059  CNX$END_FAILOVER::
     54    00000000'GF   D0      0059         MOVL    G^CLU$GL_CLUB,R4            ; Address of CLUB
       55    010C C4     9E      0060         MOVAB   CLUB$B_CLUFCB(R4),R5        ; Address of failover control block
          18 A5    27    D0      0065         MOVL    #FORCE_END, -               ; Set index to force end of table processing
                                 0069                 CLUFCB$L_STEP(R5)
                                 0069  ;
                                 0069  ; Begin a failover step
                                 0069  ;
         53    1C A5     D0      0069  40$:   MOVL    CLUFCB$L_ID(R5),R3          ; Failover ID
         50    18 A5     D0      006D         MOVL    CLUFCB$L_STEP(R5),R0        ; Get index of next failover step
   50      0000'CF40     DE      0071         MOVAL   W^CNX$FAILOVER_TABLE[R0],R0 ; Get table entry
                                 0077  ;
                                 0077  ;      R5:     Address of failover control block (with available fork block at head
                                 0077  ;      R4:     Address of CLUB
                                 0077  ;      R3:     Failover ID
                                 0077  ;
              00 B040    16      0077         JSB     @(R0)[R0]                   ; Convert self relative to absolute address
     54    00000000'GF   D0      007B         MOVL    G^CLU$GL_CLUB,R4            ; Address of CLUB
       55    010C C4     9E      0082         MOVAB   CLUB$B_CLUFCB(R4),R5        ; Address of failover control block
                                 0087  ;
                                 0087  ; Use the bits CLUFCB$V_ACTIVE, CLUFDB$V_PENDING, and CLUB$V_LOST_CNX to decide whet
                                 0087  ; to continue this failover, start again at the beginning of the table, or drop ever
                                 0087  ; knowing that a new failover will begin soon (the last case is assured if a connect
                                 0087  ; to a cluster member is broken).
                                 0087  ;
         0A 20 A5  00    E0      0087         BBS     #CLUFCB$V_ACTIVE, -         ; Branch if new failover may not
                                 008C                 CLUFCB$L_STATUS(R5),50$ ;  commence here
         0A 20 A5  01    E0      008C         BBS     #CLUFCB$V_PENDING, -        ; Branch if another failover is pending
                                 0091                 CLUFCB$L_STATUS(R5),60$ ;  and initiate it
         08 1C A4  17    E0      0091         BBS     #CLUB$V_LOST_CNX, -         ; Branch if a connection has been lost and
                                 0096                 CLUB$L_FLAGS(R4),70$    ;  and new failover will soon happen
                18 A5    D6      0096  50$:   INCL    CLUFCB$L_STEP(R5)           ; Advance to next failover routine
                   CE    11      0099         BRB     40$                         ; Do next step
                                 009B
                 FF62    31      009B  60$:   BRW     1$                          ; Start a new failover
                                 009E
```

```
05  009E    478 70$:    RSB                             ; Drop thread -- another failover will come
    009F    479
    009F    480         .DISABLE        LSB
```

```
                    009F            .SBTTL  CNX$FAILO_REFAIL - Allow new failovers to start
                    009F
                    009F    ;++
                    009F    ;
                    009F    ; FUNCTIONAL DESCRIPTION:
                    009F    ;
                    009F    ;       Enter REFAIL mode in which new executions of the failover
                    009F    ;       table may be initiated.
                    009F    ;
                    009F    ; CALLING SEQUENCE:
                    009F    ;
                    009F    ;       JSB     CNX$FAILO_REFAIL
                    009F    ;       IPL is IPL$_SCS
                    009F    ;
                    009F    ; INPUT PARAMETERS:
                    009F    ;
                    009F    ;       R5:     Address of Failover Control Block
                    009F    ;       R4:     Address of CLUB
                    009F    ;       R3:     Failover ID
                    009F    ;
                    009F    ; OUTPUT PARAMETERS:
                    009F    ;
                    009F    ;       NONE
                    009F    ;
                    009F    ; COMPLETION CODES:
                    009F    ;
                    009F    ;       NONE
                    009F    ;
                    009F    ; SIDE EFFECTS:
                    009F    ;
                    009F    ;       NONE
                    009F    ;
                    009F    ;--
                    009F
                    009F    CNX$FAILO_REFAIL::
 20 A5    01    CA  009F            BICL2   #CLUFCB$M_ACTIVE, -     ; Clear active bit
                    00A3                    CLUFCB$L_STATUS(R5)
                05  00A3            RSB
```

```
                           00A4          .SBTTL  CNX$FAILO_NOREFAIL - Prevent second entrance to failover table
                           00A4
                           00A4     ;++
                           00A4     ;
                           00A4     ; FUNCTIONAL DESCRIPTION:
                           00A4     ;
                           00A4     ;
                           00A4     ;        Stop allowing new failovers to begin.
                           00A4     ;        Terminate any previous failover that may have been in progress.
                           00A4     ;
                           00A4     ; CALLING SEQUENCE:
                           00A4     ;
                           00A4     ;        JSB     CNX$FAILO_NOREFAIL
                           00A4     ;        IPL is IPL$_SCS
                           00A4     ;
                           00A4     ; INPUT PARAMETERS:
                           00A4     ;
                           00A4     ;        R5:     Address of Failover Control Block
                           00A4     ;        R4:     Address of CLUB
                           00A4     ;        R3:     Failover ID
                           00A4     ;
                           00A4     ; OUTPUT PARAMETERS:
                           00A4     ;
                           00A4     ;        NONE
                           00A4     ;
                           00A4     ; COMPLETION CODES:
                           00A4     ;
                           00A4     ;        NONE
                           00A4     ;
                           00A4     ; SIDE EFFECTS:
                           00A4     ;
                           00A4     ;        NONE
                           00A4     ;
                           00A4     ;--
                           00A4
                           00A4     CNX$FAILO_NOREFAIL::
0C 20 A5    00    E2       00A4              BBSS    #CLUFCB$V_ACTIVE, -          ; Set active bit and branch
                           00A9                      CLUFCB$L_STATUS(R5),10$ ;      if it was already set
07 20 A5    03    E5       00A9              BBCC    #CLUFCB$V_FKB_BUSY, -        ; Branch if fork block free
                           00AE                      CLUFCB$L_STATUS(R5),10$ ;      and mark it free
      50    65    0F       00AE              REMQUE  (R5),R0                      ; Remove from queue
   20 A5    10    CA       00B1              BICL2   #CLUFCB$M_WAITING, -         ; Clear waiting bit
                           00B5                      CLUFCB$L_STATUS(R5)
                     05    00B5     10$:     RSB
```

```
      00B6             .SBTTL  CNX$FAILO_NOP - Null failover routine
      00B6
      00B6    ;++
      00B6    ;
      00B6    ; FUNCTIONAL DESCRIPTION:
      00B6    ;
      00B6    ;       Null failover routine.  A legitimate failover routine that does
      00B6    ;       absolutely nothing.
      00B6    ;
      00B6    ; CALLING SEQUENCE:
      00B6    ;
      00B6    ;       JSB     CNX$FAILO_NOP
      00B6    ;       IPL is IPL$_SCS
      00B6    ;
      00B6    ; INPUT PARAMETERS:
      00B6    ;
      00B6    ;       R5:     Address of Failover Control Block
      00B6    ;       R4:     Address of CLUB
      00B6    ;       R3:     Failover ID
      00B6    ;
      00B6    ; OUTPUT PARAMETERS:
      00B6    ;
      00B6    ;       NONE
      00B6    ;
      00B6    ; COMPLETION CODES:
      00B6    ;
      00B6    ;       NONE
      00B6    ;
      00B6    ; SIDE EFFECTS:
      00B6    ;
      00B6    ;       NONE
      00B6    ;
      00B6    ;--
      00B6
      00B6    CNX$FAILO_NOP::
   05 00B6            RSB
```

```
                   00B7           .SBTTL  CNX$FAILO_END - End failover table processing
                   00B7
                   00B7  ;++
                   00B7  ;
                   00B7  ; FUNCTIONAL DESCRIPTION:
                   00B7  ;
                   00B7  ;          End failover table processing.  This assumes that REFAIL
                   00B7  ;          mode is in effect.
                   00B7  ;
                   00B7  ; CALLING SEQUENCE:
                   00B7  ;
                   00B7  ;          JSB     CNX$FAILO_END
                   00B7  ;          IPL is IPL$_SCS
                   00B7  ;
                   00B7  ; INPUT PARAMETERS:
                   00B7  ;
                   00B7  ;          R5:     Address of Failover Control Block
                   00B7  ;          R4:     Address of CLUB
                   00B7  ;          R3:     Failover ID
                   00B7  ;
                   00B7  ; OUTPUT PARAMETERS:
                   00B7  ;
                   00B7  ;          NONE
                   00B7  ;
                   00B7  ; COMPLETION CODES:
                   00B7  ;
                   00B7  ;          NONE
                   00B7  ;
                   00B7  ; SIDE EFFECTS:
                   00B7  ;
                   00B7  ;          NONE
                   00B7  ;
                   00B7  ;--
                   00B7
                   00B7  CNX$FAILO_END::
5E    04    C0     00B7           ADDL2   #4,SP                     ; Remove caller's address
            05     00BA           RSB                               ; Return to caller's caller
                   00BB
```

```
                        00BB        .SBTTL  CNX$IO_SYNCH - Synchronize with I/O cluster
                        00BB
                        00BB  ;++
                        00BB  ;
                        00BB  ; FUNCTIONAL DESCRIPTION:
                        00BB  ;
                        00BB  ;       Synchronize with the I/O cluster.
                        00BB  ;
                        00BB  ;       Serialize I/O to multi-host disks to ensure that I/O issued under current lo
                        00BB  ;       completed before I/O issued under locks granted after some locks have been r
                        00BB  ;       as the result of removing nodes from the cluster.
                        00BB  ;
                        00BB  ;       Note that it is unnecessary to do this operation if nodes are being added an
                        00BB  ;       are being removed.
                        00BB  ;
                        00BB  ;       This entry is called following a synchronization call to ensure that all nod
                        00BB  ;       have disconnected from a node being removed before this call is made.  It is
                        00BB  ;       important that the node being removed has either lost quorum or crashed and
                        00BB  ;       therefore ceased to issue I/O's before this call is made.
                        00BB  ;
                        00BB  ; CALLING SEQUENCE:
                        00BB  ;
                        00BB  ;       JSB     CNX$IO_SYNCH
                        00BB  ;       IPL is IPL$_SCS
                        00BB  ;
                        00BB  ; INPUT PARAMETERS:
                        00BB  ;
                        00BB  ;       R5:     Address of Failover Control Block
                        00BB  ;       R4:     Address of CLUB
                        00BB  ;       R3:     Failover ID
                        00BB  ;
                        00BB  ; OUTPUT PARAMETERS:
                        00BB  ;
                        00BB  ;       NONE
                        00BB  ;
                        00BB  ; COMPLETION CODES:
                        00BB  ;
                        00BB  ;       NONE
                        00BB  ;
                        00BB  ; SIDE EFFECTS:
                        00BB  ;
                        00BB  ;       Disks are thrown into mount verification.
                        00BB  ;
                        00BB  ;--
                        00BB
                        00BB  CNX$IO_SYNCH:
00000000'GF     17      00BB        JMP     G^EXE$CLUTRANIO         ; Synchronize with I/O cluster and return
                        00C1
```

```
                        00C1         .SBTTL  CNX$FAILO_SYNC - Inter-Node Synchronization Failover Routine
                        00C1
                        00C1  ;++
                        00C1  ;
                        00C1  ; FUNCTIONAL DESCRIPTION:
                        00C1  ;
                        00C1  ;      This routine synchronizes failover action over all involved nodes.
                        00C1  ;      This failover routine does not complete until all nodes have completed
                        00C1  ;      the previous failover step.
                        00C1  ;      A new failover may be begun while this routine is in progress.
                        00C1  ;
                        00C1  ;      The synchronization algorithm is a follows:
                        00C1  ;
                        00C1  ;      One node is chosen as the node to drive the synchronization.  All other
                        00C1  ;      nodes send a message to the sync node when this routine is executed.
                        00C1  ;      If the sync node NAKs the message, the node waits and tries again.  If
                        00C1  ;      the message fails, the node drops its thread, confident that a new
                        00C1  ;      failover will begin sometime and start the whole thing over.  If the
                        00C1  ;      message succeeds, the node then waits for a DOSTEP message, upon
                        00C1  ;      receipt of which it exits from this failover step.
                        00C1  ;
                        00C1  ;      Meanwhile, the sync node waits for all other nodes to report in.  When
                        00C1  ;      a message is received for the right failover, it is noted and ACKed.
                        00C1  ;      If a message is not for the current failover, it is NAKed.  After all
                        00C1  ;      nodes has reported, a DOSTEP message is sent to every other node.
                        00C1  ;      When a connection breaks the CNX$CON_BREAK routine sets a bit that
                        00C1  ;      eliminates that node from consideration by this routine.
                        00C1  ;
                        00C1  ; CALLING SEQUENCE:
                        00C1  ;
                        00C1  ;      JSB     CNX$FAILO_SYNC
                        00C1  ;      IPL is IPL$_SCS
                        00C1  ;
                        00C1  ; INPUT PARAMETERS:
                        00C1  ;
                        00C1  ;      R5:     Address of Failover Control Block
                        00C1  ;      R4:     Address of CLUB
                        00C1  ;      R3:     Failover ID
                        00C1  ;
                        00C1  ; OUTPUT PARAMETERS:
                        00C1  ;
                        00C1  ;      NONE
                        00C1  ;
                        00C1  ; COMPLETION CODES:
                        00C1  ;
                        00C1  ;      NONE
                        00C1  ;
                        00C1  ; SIDE EFFECTS:
                        00C1  ;
                        00C1  ;      NONE
                        00C1  ;
                        00C1  ;--
                        00C1
                        00C1  CNX$FAILO_SYNC::
03 20 A5    02    E0    00C1         BBS     #CLUFCB$V_SYNC_NODE, -  ; Branch if this is the
                        00C6                 CLUFCB$L_STATUS(R5), -  ;   synchronizing node
                        00C6                 5$
```

```
                      0098  31  00C6            BRW         100$                        ; Branch to code for other nodes
                                00C9
                50  60 A4   3C  00C9  5$:       MOVZWL      CLUB$W_LOCAL_CSID_IDX(R4),R0 ; Local CSID index field
             00 48 A5   50  E3  00CD            BBCS        R0, -                       ; Mark local node responded
                                00D2                        CLUFCB$B_RESPMAP(R5),7$
                        0B  11  00D2  7$:       BRB         14$                         ; Branch to base of loop
                                00D4
                20 A5   10  C8  00D4  10$:      BISL2       #CLUFCB$M_WAITING, -        ; Set waiting bit
                                00D8                        CLUFCB$L_STATUS(R5)
                      0103  30  00D8            BSBW        DELAY                       ; Wait for 0-1 second
                20 A5   10  CA  00DB            BICL2       #CLUFCB$M_WAITING, -        ; Clear waiting bit
                                00DF                        CLUFCB$L_STATUS(R5)
                   51   1F  D0  00DF  14$:      MOVL        #CLUFCB$S_RESPMAP-1,R1      ; Byte counter
    50   28 A541    48 A541  8B  00E2  15$:      BICB3       CLUFCB$B_RESPMAP(R5)[R1], - ; See if all currently involved
                                00EA                        CLUFCB$B_NODEMAP(R5)[R1], - ;   nodes have responded
                                00EA                        R0
                        E8  12  00EA            BNEQ        10$                         ; Found unresponded node(s)
                     F3 51  F4  00EC            SOBGEQ      R1,15$                      ; Iterate over all nodes
                        55  DD  00EF            PUSHL       R5                          ; Save register
  48 A5   20    00   6E  00  2C  00F1            MOVC5       #0,(SP),#0, -               ; Zero out RESPMAP for next use
                                00F8                        #CLUFCB$S_RESPMAP, -
                                00F8                        CLUFCB$B_RESPMAP(R5)
                        20  BA  00F8            POPR        #^M<R5>                     ; Restore register
                                00FA      ;
                                00FA      ; All reponses have been received.  Send a message to every
                                00FA      ; node telling it to proceed with the next step.
                                00FA      ;
             54    00FF 8F  3C  00FA            MOVZWL      #<CLUFCB$S_NODEMAP*8>-1,R4  ; Index in bitmap
                        09  11  00FF            BRB         30$                         ; Branch to loop entrance
                                0101
                                0101  19$:      BUG_CHECK         CNXMGRERR,FATAL      ; Consistency check
                                0105
                        30  BA  0105  20$:      POPR        #^M<R4,R5>                  ; Restore index and CLUFCB addresses
                      00D4  30  0107            BSBW        DELAY                       ; Wait 0-1 seconds
             04 28 A5   54  E0  010A  30$:      BBS         R4,CLUFCB$B_NODEMAP(R5),50$ ; Branch if in map
                     F8 54  F4  010F  40$:      SOBGEQ      R4,30$                      ; Iterate over all possible bits
                        05      0112            RSB                                     ; All done, return to caller
                                0113
          51    00000000'GF  D0  0113  50$:      MOVL        G^CLU$GL_CLUSVEC,R1         ; Address of cluster vector
                53    6144  D0  011A            MOVL        (R1)[R4],R3                 ; Address of CSB
                        E1  18  011E            BGEQ        19$                         ; Not a CSB -- consistency check
             EA 60 A3   18  E0  0120            BBS         #CSB$V_LOCAL, -             ; Branch if local node to avoid
                                0125                        CSB$L_STATUS(R3),40$        ;   trying to send message to self
                        30  BB  0125            PUSHR       #^M<R4,R5>                  ; Save context
                     FED6'  30  0127            BSBW        CNX$ALLOC_CDRP_ONLY         ; Get a fork block
                     FED3'  30  012A            BSBW        CNX$RESOURCE_CHECK          ; Watch resource availability
                     D5 50  E9  012D            BLBC        R0,20$                      ; Branch if no memory available
                54  04 AE   D0  0130            MOVL        4(SP),R4                    ; Restore CLUFCB address
                2C A5   0C  9A  0134            MOVZBL      #CLMCNX$K_FNC_DOSTEP, -     ; Function code for message
                                0138                        CDRP$L_VA[1(R5)
                      0093  30  0138            BSBW        300$                        ; Initialize CDRP for mesage building
                     FEC2'  30  013B            BSBW        CNX$SEND_FORGET             ; Send message to remote node and return imm
                        30  BA  013E            POPR        #^M<R4,R5>                  ; Restore index and CLUFCB addresses
             BC 20 A5   03  E2  0140            BBSS        #CLUFCB$V_FKB_BUSY, -       ; Branch if busy and mark busy
                                0145                        CLUFCB$L_STATUS(R5),19$
                0B A5   08  90  0145            MOVB        #IPL$_SCS, -                ; Store IPL in fork block
                                0149                        FKB$B_FIPL(R5)
```

```
             08  BA  0149          POPR    #^M<R3>                  ; Save caller's address
      00000000'GF  16  014B        JSB     G^EXE$FORK               ; Release control momentarily
             53  DD  0151          PUSHL   R3                       ; Restore caller's address
   20 A5     08  CA  0153          BICL2   #CLUFCB$M_FKB_BUSY, -    ; Clear busy bit
                     0157                  CLUFCB$L_STATUS(R5)
B3 20 A5     01  E1  0157          BBC     #CLUFCB$V_PENDING, -     ; Branch if no failover is
                     015C                  CLUFCB$L_STATUS(R5),40$  ;  pending
                 05  015C          RSB                              ; Return - new failover is pending
                     015D
                     015D   ;
                     015D   ; This concludes the code that executes in the synchronizing node.
                     015D   ; Here begins the code that executes in the non-synchronizing nodes.
                     015D   ;
             20  BA  015D   90$:   POPR    #^M<R5>                  ; Restore CLUFCB addresses
             7D  10  015F   95$:   BSBB    DELAY                    ; Wait 0-1 seconds
                     0161   ;
                     0161   ; Send message to synchronizing node telling it that we are ready for
                     0161   ; a command to finish a failover step.
                     0161   ;
   53  24 A5     D0  0161   100$:  MOVL    CLUFCB$L_SYNC_CSB(R5),R3 ; CSB address of synchronizing node
             55  DD  0165          PUSHL   R5                       ; Save context
          FE96'  30  0167          BSBW    CNX$ALLOC_WARMCDRP_CSB   ; Get a fork block
          FE93'  30  016A          BSBW    CNX$RESOURCE_CHECK       ; Check for pool exhaustion
          ED 50  E9  016D          BLBC    R0,90$                   ; Branch if no memory available
             10  BA  0170          POPR    #^M<R4>                  ; Restore CLUFCB address into R4
   40 A5     54  D0  0172          MOVL    R4,CDRP$L_VAL6(R5)       ; Save CLUFCB address
      44 A5  8ED0    0176          POPL    CDRP$L_VAL7(R5)          ; Completion address
   2C A5     0B  9A  017A          MOVZBL  #CLMCNX$K_FNC_READY, -   ; Function code for message
                     017E                  CDRP$L_VAL1(R5)
             4E  10  017E          BSBB    300$                     ; Set up CDRP for message
          FE7D'  30  0180          BSBW    CNX$SEND_MSG_CSB         ; Send message to remote node
                     0183   ;
                     0183   ;      Resume here when the response message arrives or the connection
                     0183   ;      breaks.
                     0183   ;      Registers contain:
                     0183   ;              R0:     Status
                     0183   ;              R2:     Address of message buffer
                     0183   ;              R3:     Address of CSB
                     0183   ;              R4:     Address of PDT
                     0183   ;              R5:     Address of CDRP
                     0183   ;
          44 A5  DD  0183          PUSHL   CDRP$L_VAL7(R5)          ; Completion address
          30 A5  DD  0186          PUSHL   CDRP$L_VAL2(R5)          ; Failover sequence number
          40 A5  DD  0189          PUSHL   CDRP$L_VAL6(R5)          ; CLUFCB address
          FE71'  30  018C          BSBW    CNX$PROCESS_RESPONSE     ; Standard response processing
             20  BA  018F          POPR    #^M<R5>                  ; CLUFCB address
   1C A5     8E  D1  0191          CMPL    (SP)+,CLUFCB$L_ID(R5)    ; Same step?
             23  12  0195          BNEQ    110$                     ; Abandon this failover
      01     50  D1  0197          CMPL    R0,#1                    ; Test return status
             C3  1F  019A          BLSSU   95$                      ; Branch on NAK
             1C  1A  019C          BGTRU   110$                     ; Connection is broken
                     019E   ;
                     019E   ; ACK received
                     019E   ;
27 20 A5     03  E2  019E          BBSS    #CLUFCB$V_FKB_BUSY, -    ; Branch if busy and mark busy
                     01A3                  CLUFCB$L_STATUS(R5),140$
   20 A5     10  C8  01A3          BISL2   #CLUFCB$M_WAITING, -     ; Set waiting bit, error if
```

```
                          01A7                    CLUFCB$L_STATUS(R5)       ;  already set
            53   6E   D0  01A7           MOVL     (SP),R3                   ; Save return PC
                          01AA           ASSUME   FKB$L_FR4,EQ,<FKB$L_FR3+4>
      10 A5      53   7D  01AA           MOVQ     R3,FKB$L_FR3(R5)          ; Save R3 and R4
   0C A5    BE'AF     9E  01AE           MOVAB    B^120$,FKB$L_FPC(R5)      ; Restart PC
         65      55   D0  01B3           MOVL     R5,(R5)                   ; Link to self
      04 A5      55   D0  01B6           MOVL     R5,4(R5)
         5E      04   C0  01BA   110$:   ADDL2    #4,SP                     ; Forget return address
                      05  01BD           RSB
                          01BE
                 53   DD  01BE   120$:   PUSHL    R3                        ; Restore caller's address
   05 20 A5      03   E5  01C0           BBCC     #CLUFCB$V_FKB_BUSY, -     ; Clear busy flag
                          01C5                    CLUFCB$L_STATUS(R5),140$
      20 A5      10   CA  01C5           BICL2    #CLUFCB$M_WAITING, -      ; Clear waiting bit
                          01C9                    CLUFCB$L_STATUS(R5)
                      05  01C9           RSB
                          01CA
                          01CA   140$:   BUG_CHECK        CNXMGRERR,FATAL ; Consistency check
                          01CE
                          01CE   ;
                          01CE   ;       Initialize CDRP for building messages
                          01CE   ;
                          01CE   ;       R5:     Address of CDRP
                          01CE   ;       R4:     Address of CLUFCB
                          01CE   ;
   4C A5    04'AF     9E  01CE   300$:   MOVAB    B^BLD_STEP_MSG, -         ; Message build routine address
                          01D3                    CDRP$L_MSGBLD(R5)
   30 A5    1C A4     D0  01D3           MOVL     CLUFCB$L_ID(R4), -        ; Failover sequence number
                          01D8                    CDRP$L_VAL2(R5)
   34 A5    18 A4     D0  01D8           MOVL     CLUFCB$L_STEP(R4), -      ; Failover routine index
                          01DD                    CDRP$L_VAL3(R5)
                      05  01DD           RSB
                          01DE
                          01DE   ;
                          01DE   ;       Wait 0-1 second on fork and wait queue
                          01DE   ;       If a new failover is requested, return to caller's caller
                          01DE   ;
                          01DE   ;       R5:     Address of CLUFCB fork block
                          01DE   ;       R0-R4 are destroyed.
                          01DE   ;
                          01DE   DELAY:
   1D 20 A5      03   E2  01DE           BBSS     #CLUFCB$V_FKB_BUSY, -     ; Branch if busy and mark busy
                          01E3                    CLUFCB$L_STATUS(R5),20$
      0B A5      08   90  01E3           MOVB     #IPL$_SCS, -              ; Store IPL in fork block
                          01E7                    FKB$B_FIPL(R5)
                 10   BA  01E7           POPR     #^M<R4>                   ; Save return address
                 08   BA  01E9           POPR     #^M<R3>                   ; Save caller's caller's address
                          01EB           FORK_WAIT                          ; Wait and hope some responses appear
                 53   DD  01F1           PUSHL    R3                        ; Restore caller's caller's address
   08 20 A5      03   E5  01F3           BBCC     #CLUFCB$V_FKB_BUSY, -     ; Clear busy bit
                          01F8                    CLUFCB$L_STATUS(R5),20$
   02 20 A5      01   E0  01F8           BBS      #CLUFCB$V_PENDING, -      ; Branch if another failover is
                          01FD                    CLUFCB$L_STATUS(R5),10$ ;   pending and do it
                 54   DD  01FD           PUSHL    R4                        ; Restore return address
                      05  01FF   10$:    RSB
                          0200
                          0200   20$:    BUG_CHECK        CNXMGRERR,FATAL ; Consistency check
```

```
                      0204          .SBTTL  BLD_STEP_MSG - Build Message to Step Failover
                      0204
                      0204  ;++
                      0204  ;
                      0204  ; FUNCTIONAL DESCRIPTION:
                      0204  ;
                      0204  ;       This routine builds the messages used to sequence failover steps
                      0204  ;       from data in the CDRP.
                      0204  ;
                      0204  ; CALLING SEQUENCE:
                      0204  ;
                      0204  ;       JSB     BLD_STEP_MSG
                      0204  ;       IPL is IPL$_SCS
                      0204  ;
                      0204  ; INPUT PARAMETERS:
                      0204  ;
                      0204  ;       R2:     Address of message buffer
                      0204  ;       R3:     Address of CSB
                      0204  ;       R4:     Address of PDT
                      0204  ;       R5:     Address of CDRP
                      0204  ;               CDRP$L_VAL1(R5): Byte 0 contains facility specific function code
                      0204  ;               CDRP$L_VAL2(R5): Failover sequence number
                      0204  ;               CDRP$L_VAL3(R5): Failover step index
                      0204  ;
                      0204  ; OUTPUT PARAMETERS:
                      0204  ;
                      0204  ;       NONE
                      0204  ;
                      0204  ; COMPLETION CODES:
                      0204  ;
                      0204  ;       NONE
                      0204  ;
                      0204  ; SIDE EFFECTS:
                      0204  ;
                      0204  ;       R0 and R1 are destroyed
                      0204  ;
                      0204  ;--
                      0204
                      0204  BLD_STEP_MSG:
  08 A2     01   90   0204          MOVB    #CLSMSG$K_FAC_CNX, -    ; Facility code
                      0208                  CLSMSG$B_FACILITY(R2)
  09 A2  2C A5   90   0208          MOVB    CDRP$L_VAL1(R5), -      ; Facility specific function code
                      020D                  CLSMSG$B_FUNC(R2)
  0C A2  30 A5   D0   020D          MOVL    CDRP$L_VAL2(R5), -      ; Failover sequence number
                      0212                  CLMSTP$L_ID(R2)
  10 A2  34 A5   D0   0212          MOVL    CDRP$L_VAL3(R5), -      ; Failover step index
                      0217                  CLMSTP$L_STEP(R2)
                 05   0217          RSB
```

```
                        0218          .SBTTL  CNX$RCVD_READY - Ready for Failover Step Message Received
                        0218
                        0218  ;++
                        0218  ;
                        0218  ; FUNCTIONAL DESCRIPTION:
                        0218  ;
                        0218  ;       This routine is called when a message from a node ready to perform
                        0218  ;       a failover step is received.
                        0218  ;
                        0218  ; CALLING SEQUENCE:
                        0218  ;
                        0218  ;       JSB     CNX$RCVD_READY
                        0218  ;       IPL is IPL$_SCS
                        0218  ;
                        0218  ; INPUT PARAMETERS:
                        0218  ;
                        0218  ;       R2:     Message address
                        0218  ;       R3:     CSB of sending system
                        0218  ;       R4:     PDT address
                        0218  ;       R5:     CDRP address (uninitialized)
                        0218  ;
                        0218  ; OUTPUT PARAMETERS:
                        0218  ;
                        0218  ;       NONE
                        0218  ;
                        0218  ; COMPLETION CODES:
                        0218  ;
                        0218  ;       NONE
                        0218  ;
                        0218  ; SIDE EFFECTS:
                        0218  ;
                        0218  ;       R0-R5 may be destroyed.
                        0218  ;--
                        0218
                        0218  CNX$RCVD_READY::
             52    DD   0218          PUSHL   R2                              ; Message buffer address
           FDE3'   30   021A          BSBW    CNX$INIT_CDRP                   ; Initialize the CDRP for the response
             04    BA   021D          POPR    #^M<R2>                         ; Restore message buffer address
       54  64 A3   D0   021F          MOVL    CSB$L_CLUB(R3),R4               ; Address of CLUB
   54    010C C4   9E   0223          MOVAB   CLUB$B_CLUFCB(R4),R4            ; Address of failover block
             50    D4   0228          CLRL    R0                              ; Assume failure
 1C A4    0C A2    D1   022A          CMPL    CLMSTP$L_ID(R2), -              ; Is the expected failover in progress?
                        022F                  CLUFCB$L_ID(R4)
             18    12   022F          BNEQ    20$                             ; Branch if not and ignore message
 18 A4    10 A2    D1   0231          CMPL    CLMSTP$L_STEP(R2), -            ; Has this step been passed?
                        0236                  CLUFCB$L_STEP(R4)
             39    1F   0236          BLSSU   90$                             ; Branch if yes -- something is wrong
 34 20 A4     02   E1   0238          BBC     #CLUFCB$V_SYNC_NODE, -          ; Branch if this is not the synchronizing
                        023D                  CLUFCB$L_STATUS(R4),90$         ;   node
         50   01   D0   023D          MOVL    #1,R0                           ; Set success
       51  4C A3   3C   0240          MOVZWL  CSB$W_CSID_IDX(R3),R1           ; CSID index of sending system
 28 48 A4     51   E2   0244          BBSS    R1, -                           ; Set the response bit corresponding
                        0249                  CLUFCB$B_RESPMAP(R4),90$        ;   to the sending node
             11    BB   0249  20$:    PUSHR   #^M<R0,R4>                      ; Save status and CLUFCB address
       54  64 A3   D0   024B          MOVL    CSB$L_CLUB(R3),R4               ; Address of CLUB
         50   0B   9A   024F          MOVZBL  #CLMCNX$K_FNC_READY,R0          ; Message ID
           FDAB'   30   0252          BSBW    CNX$INIT_STD_RESP               ; Set up response message
```

```
          36 A5    6E    90 0255            MOVB    (SP),CDRP$L_VAL3+2(R5)  ; Store status flag
                FDA4'    30 0259            BSBW    CNX$RESP_FORGET         ; Send response and exit
                   21    BA 025C            POPR    #^M<R0,R5>              ; Restore status and CLUFCB address
               0F  50    E9 025E            BLBC    R0,30$                  ; Branch if message NAKed
       0A 20 A5    04    E1 0261            BBC     #CLUFCB$V_WAITING, -    ; Branch if not waiting for responses
                            0266                    CLUFCB$L_STATUS(R5),30$
                            0266            ;
                            0266            ; Synchronizing node waiting for nodes to respond.
                            0266            ; Wake it up.
                            0266            ;
             55    65    0F 0266            REMQUE  (R5),R5                 ; Dequeue fork block
                            0269            ASSUME  FKB$L_FR4,EQ,<FKB$L_FR3+4>
          53    10 A5    7D 0269            MOVQ    FKB$L_FR3(R5),R3        ; Restore R3 and R4
                0C B5    16 026D            JSB     @FKB$L_FPC(R5)          ; Call fork routine
                         05 0270      30$:  RSB                             ; Return
                            0271
                            0271      90$:  BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
                          0275          .SBTTL  CNX$RCVD_DOSTEP - Do Failover Step Message Received
                          0275
                          0275  ;++
                          0275  ;
                          0275  ; FUNCTIONAL DESCRIPTION:
                          0275  ;
                          0275  ;       This routine is called when a message from the synchronizing node is
                          0275  ;       received requesting that a failover step be performed.
                          0275  ;
                          0275  ; CALLING SEQUENCE:
                          0275  ;
                          0275  ;       JSB     CNX$RCVD_DOSTEP
                          0275  ;       IPL is IPL$_SCS
                          0275  ;
                          0275  ; INPUT PARAMETERS:
                          0275  ;
                          0275  ;       R2:     Message address
                          0275  ;       R3:     CSB of sending system
                          0275  ;       R4:     PDT address
                          0275  ;
                          0275  ; OUTPUT PARAMETERS:
                          0275  ;
                          0275  ;       NONE
                          0275  ;
                          0275  ; COMPLETION CODES:
                          0275  ;
                          0275  ;       NONE
                          0275  ;
                          0275  ; SIDE EFFECTS:
                          0275  ;
                          0275  ;       R0-R5 may be destroyed.
                          0275  ;--
                          0275
                          0275  CNX$RCVD_DOSTEP::
           0C A2     DD   0275          PUSHL   CLMSTP$L_ID(R2)             ; Failover sequence number
           10 A2     DD   0278          PUSHL   CLMSTP$L_STEP(R2)           ; Failover step index
           FD82'     30   027B          BSBW    CNX$DEALL_MSG_BUF_CSB       ; Deallocate message buffer
  54       64 A3     D0   027E          MOVL    CSB$L_CLUB(R3),R4           ; Address of CLUB
  55     010C C4     9E   0282          MOVAB   CLUB$B_CLUFCB(R4),R5        ; Address of failover block
              03     BA   0287          POPR    #^M<R0,R1>                  ; SEQ to R1, INDEX to R0
     1C A5    51     D1   0289          CMPL    R1,CLUFCB$L_ID(R5)          ; Is the expected failover in progress?
              17     12   028D          BNEQ    20$                         ; Branch if not and ignore message
     18 A5    50     D1   028F          CMPL    R0,CLUFCB$L_STEP(R5)        ; Check step number consistency
              12     12   0293          BNEQ    90$                         ; Branch if different -- something is wrong
  0D 20 A5    02     E0   0295          BBS     #CLUFCB$V_SYNC_NODE, -      ; Branch if this is the synchronizing
                          029A                  CLUFCB$L_STATUS(R5),90$     ;   node -- can't happen
  08 20 A5    04     E1   029A          BBC     #CLUFCB$V_WAITING, -        ; Branch if this node not waiting for respon
                          029F                  CLUFCB$L_STATUS(R5),90$
                          029F          ASSUME  FKB$L_FR4,EQ,<FKB$L_FR3+4>
     53    10 A5     7D   029F          MOVQ    FKB$L_FR3(R5),R3            ; Resume fork block
           0C B5     16   02A3          JSB     @FKB$L_FPC(R5)              ;   and continue table
              05          02A6  20$:    RSB                                 ; Return
                          02A7
                          02A7  90$:    BUG_CHECK       CNXMGRERR,FATAL     ; Consistency check
```

```
                  02AB            .SBTTL  CNX$FAILO_JNL - Initiate journal recovery
                  02AB
                  02AB      ;++
                  02AB      ;
                  02AB      ; FUNCTIONAL DESCRIPTION:
                  02AB      ;
                  02AB      ;       This routine find and invokes the journal recovery routine.
                  02AB      ;
                  02AB      ; CALLING SEQUENCE:
                  02AB      ;
                  02AB      ;       JSB     CNX$FAILO_JNL
                  02AB      ;       IPL is IPL$_SCS
                  02AB      ;
                  02AB      ; INPUT PARAMETERS:
                  02AB      ;
                  02AB      ;       R5:     Address of Failover Control Block
                  02AB      ;       R4:     Address of CLUB
                  02AB      ;       R3:     Failover ID
                  02AB      ;
                  02AB      ; OUTPUT PARAMETERS:
                  02AB      ;
                  02AB      ;       NONE
                  02AB      ;
                  02AB      ; COMPLETION CODES:
                  02AB      ;
                  02AB      ;       NONE
                  02AB      ;
                  02AB      ; SIDE EFFECTS:
                  02AB      ;
                  02AB      ;       NONE
                  02AB      ;
                  02AB      ;--
                  02AB
                  02AB      CNX$FAILO_JNL::
 50   18 A4   D0  02AB              MOVL    CLUB$L_JNL_FAIL(R4),R0  ; Fetch and test routine address
         02   13  02AF              BEQL    10$                     ; No routine found, complete failover step
         60   17  02B1              JMP     (R0)                    ; Jump to routine -- it sees standard
                  02B3                                              ;   failover routine environment
                  02B3
              05  02B3      10$:    RSB
```

```
                               02B4         .SBTTL  CNX$CHECK_FAILOVER - Test for and begin pending failover
                               02B4
                               02B4  ;++
                               02B4  ;
                               02B4  ; FUNCTIONAL DESCRIPTION:
                               02B4  ;
                               02B4  ;      This routine tests for a pending failover and initiates the pending
                               02B4  ;      failover.  If no failover is pending, a check is made for a certain
                               02B4  ;      future failover.  It is is known that a failover will soon happen, the
                               02B4  ;      thread is dropped (return to caller's caller).  If no failover is
                               02B4  ;      pending and no future failover is certain, return to the caller.
                               02B4  ;      It is assumed that failover table processing is in NOREFAIL mode.
                               02B4  ;
                               02B4  ;      This routine is used by failover routines that operate in NOREFAIL
                               02B4  ;      mode to determine whether or not to continue failover processing or
                               02B4  ;      to abandon it in favor of starting over (immediately or in the near
                               02B4  ;      future).  If a decision is made to abandon processing, no return
                               02B4  ;      is made to the caller.
                               02B4  ;
                               02B4  ; CALLING SEQUENCE:
                               02B4  ;
                               02B4  ;      JSB     CNX$CHECK_FAILOVER
                               02B4  ;      IPL is IPL$_SCS
                               02B4  ;      4(SP) : Address of caller's caller
                               02B4  ;
                               02B4  ; INPUT PARAMETERS:
                               02B4  ;
                               02B4  ;      NONE
                               02B4  ;
                               02B4  ; OUTPUT PARAMETERS:
                               02B4  ;
                               02B4  ;      NONE
                               02B4  ;
                               02B4  ; COMPLETION CODES:
                               02B4  ;
                               02B4  ;      NONE
                               02B4  ;
                               02B4  ; SIDE EFFECTS:
                               02B4  ;
                               02B4  ;      If return is to caller, R0 and R1 are destroyed.
                               02B4  ;      If return is to caller's caller, R0-R5 are destroyed.
                               02B4  ;--
                               02B4
                               02B4  CNX$CHECK_FAILOVER::
 50   00000000'GF      D0      02B4          MOVL    G^CLU$GL_CLUB,R0         ; Address of CLUB
  1F 012C C0      00   E1      02BB          BBC     #CLUFCB$V_ACTIVE, -      ; Branch if in REFAIL mode and
                               02C1                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0), -
                               02C1                  30$                      ;    bugcheck
  0B 012C C0      01   E1      02C1          BBC     #CLUFCB$V_PENDING, -     ; Branch if no failover is pending
                               02C7                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0), -
                               02C7                  10$
     012C C0      01   CA      02C7          BICL2   #CLUFCB$M_ACTIVE, -      ; Clear failover active bit
                               02CC                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0)
          5E      04   C0      02CC          ADDL2   #4,SP                    ; Remove caller's return address
             FD2E      31      02CF          BRW     CNX$MEMBERSHIP_CHANGE    ; Initiate new membership change
                               02D2
  08 1C A0        17   E1      02D2  10$:    BBC     #CLUB$V_LOST_CNX, -      ; Branch if no cluster connections broken
```

```
                     02D7          CLUB$L_FLAGS(R0),20$
012C C0   01   CA    02D7   BICL2  #CLUFCB$M_ACTIVE, -      ; Clear failover active bit
                     02DC          CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0)
     5E   04   C0    02DC   ADDL2  #4,SP                    ; Return to caller's caller
               05    02DF 20$:  RSB                         ; No pending failover, return
                     02E0
                     02E0 30$:  BUG_CHECK   CNXMGRERR,FATAL ; Consistency check
                     02E4
                     02E4       .END
```

```
BLD_STEP_MSG                      00000204 R      03
BUG$_CNXMGRERR                    ********    X   03
CDRP$L_MSGBLD                   = 0000004C
CDRP$L_VAL1                     = 0000002C
CDRP$L_VAL2                     = 00000030
CDRP$L_VAL3                     = 00000034
CDRP$L_VAL6                     = 00000040
CDRP$L_VAL7                     = 00000044
CLMCNX$K_FNC_DOSTEP             = 0000000C
CLMCNX$K_FNC_READY              = 0000000B
CLMSTP$L_ID                     = 0000000C
CLMSTP$L_STEP                   = 00000010
CLSMSG$B_FACILITY               = 00000008
CLSMSG$B_FUNC                   = 00000009
CLSMSG$K_FAC_CNX                = 00000001
CLU$GL_CLUB                       ********    X   03
CLU$GL_CLUSVEC                    ********    X   03
CLUB$B_CLUFCB                   = 0000010C
CLUB$L_COORD                    = 0000005C
CLUB$L_FLAGS                    = 0000001C
CLUB$L_JNL_FAIL                 = 00000018
CLUB$L_LOCAL_CSB                = 00000010
CLUB$L_LST_XTN                  = 00000034
CLUB$V_LOST_CNX                 = 00000017
CLUB$W_LOCAL_CSID_IDX           = 00000060
CLUFCB$B_NODEMAP                = 00000028
CLUFCB$B_RESPMAP                = 00000048
CLUFCB$L_ID                     = 0000001C
CLUFCB$L_STATUS                 = 00000020
CLUFCB$L_STEP                   = 00000018
CLUFCB$L_SYNC_CSB               = 00000024
CLUFCB$M_ACTIVE                 = 00000001
CLUFCB$M_FKB_BUSY               = 00000008
CLUFCB$M_PENDING                = 00000002
CLUFCB$M_SYNC_NODE              = 00000004
CLUFCB$M_WAITING                = 00000010
CLUFCB$S_NODEMAP                = 00000020
CLUFCB$S_RESPMAP                = 00000020
CLUFCB$V_ACTIVE                 = 00000000
CLUFCB$V_FKB_BUSY               = 00000003
CLUFCB$V_PENDING                = 00000001
CLUFCB$V_SYNC_NODE              = 00000002
CLUFCB$V_WAITING                = 00000004
CNX$ALLOC_CDRP_ONLY               ********    X   03
CNX$ALLOC_WARMCDRP_CSB            ********    X   03
CNX$CHECK_FAILOVER                000002B4 RG     03
CNX$DEALL_MSG_BUF_CSB             ********    X   03
CNX$END_FAILOVER                  00000059 RG     03
CNX$FAILOVER_TABLE                00000000 RG     02
CNX$FAILO_END                     000000B7 RG     03
CNX$FAILO_JNL                     000002AB RG     03
CNX$FAILO_NOP                     000000B6 RG     03
CNX$FAILO_NOREFAIL                000000A4 RG     03
CNX$FAILO_REFAIL                  0000009F RG     03
CNX$FAILO_SYNC                    000000C1 RG     03
CNX$INIT_CDRP                     ********    X   03
CNX$INIT_STD_RESP                 ********    X   03
CNX$IO_SYNCH                      000000BB R      03
CNX$MEMBERSHIP_CHANGE             00000000 RG     03
CNX$PROCESS_RESPONSE              ********    X   03
CNX$RCVD_DOSTEP                   00000275 RG     03
CNX$RCVD_READY                    00000218 RG     03
CNX$RESOURCE_CHECK                ********    X   03
CNX$RESP_FORGET                   ********    X   03
CNX$SCAN_CSBS                     ********    X   03
CNX$SEND_FORGET                   ********    X   03
CNX$SEND_MSG_CSB                  ********    X   03
CSB$L_CLUB                      = 00000064
CSB$L_STATUS                    = 00000060
CSB$V_LOCAL                     = 00000018
CSB$V_LONG_BREAK                = 00000000
CSB$V_MEMBER                    = 00000001
CSB$W_CSID_IDX                  = 0000004C
DELAY                             000001DE R      03
EXE$CLUTRANIO                     ********    X   03
EXE$FORK                          ********    X   03
EXE$FORK_WAIT                     ********    X   03
FKB$B_FIPL                      = 0000000B
FKB$L_FPC                       = 0000000C
FKB$L_FR3                       = 00000010
FKB$L_FR4                       = 00000014
FORCE_END                       = 00000027
IPL$_SCS                        = 00000008
IPL$_SYNCH                      = 00000008
IPL$_TIMER                      = 00000008
LCK$INIT_REBUILD                  ********    X   02
LCK$REBUILD_LKBS                  ********    X   02
LCK$REBUILD_RSBS                  ********    X   02
LCK$RESUME_ALL                    ********    X   02
LCK$RESUME_UNPROT                 ********    X   02
LCK$SET_STATE0                    ********    X   02
LCK$SET_STATE1                    ********    X   02
LCK$SET_STATE2                    ********    X   02
LCK$SET_STATE3                    ********    X   02
LCK$STALL_ALL                     ********    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
$$$060                     000000C4  (   196.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG
$$$100                     000002E4  (   740.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD     WRT NOVEC LONG

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                  29    00:00:00.04    00:00:01.06
Command processing             112    00:00:00.42    00:00:02.71
Pass 1                         384    00:00:08.70    00:00:31.77
Symbol table sort                0    00:00:01.39    00:00:03.24
Pass 2                         215    00:00:02.37    00:00:08.20
Symbol table output             12    00:00:00.06    00:00:00.06
Psect synopsis output            2    00:00:00.02    00:00:00.15
Cross-reference output           0    00:00:00.00    00:00:00.00
Assembler run totals           756    00:00:13.01    00:00:47.59

The working set limit was 1650 pages.
82112 bytes (161 pages) of virtual memory were used to buffer the intermediate code.
There were 80 pages of symbol table space allocated to hold 1230 non-local and 78 local symbols.
1205 source lines were read in Pass 1, producing 19 object records in Pass 2.
18 pages of virtual memory were used to define 17 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1                    1
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                           7
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        5
TOTALS (all libraries)                                  13

1289 GETS were required to define 13 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:FALMAN/OBJ=OBJ$:FALMAN MSRC$:FALMAN/UPDATE=(ENH$:FALMAN)+EXECML$/LIB+LIB$:CLUSTER/LIB
```

0395 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

ERRSUBUV1
LIS

ERRSUB790
LIS

ERRSUB780
LIS

ERRSUB750
LIS

FALMAN
LIS

ERRSUB730
LIS

0396 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUV1
LIS

INIADP790
LIS