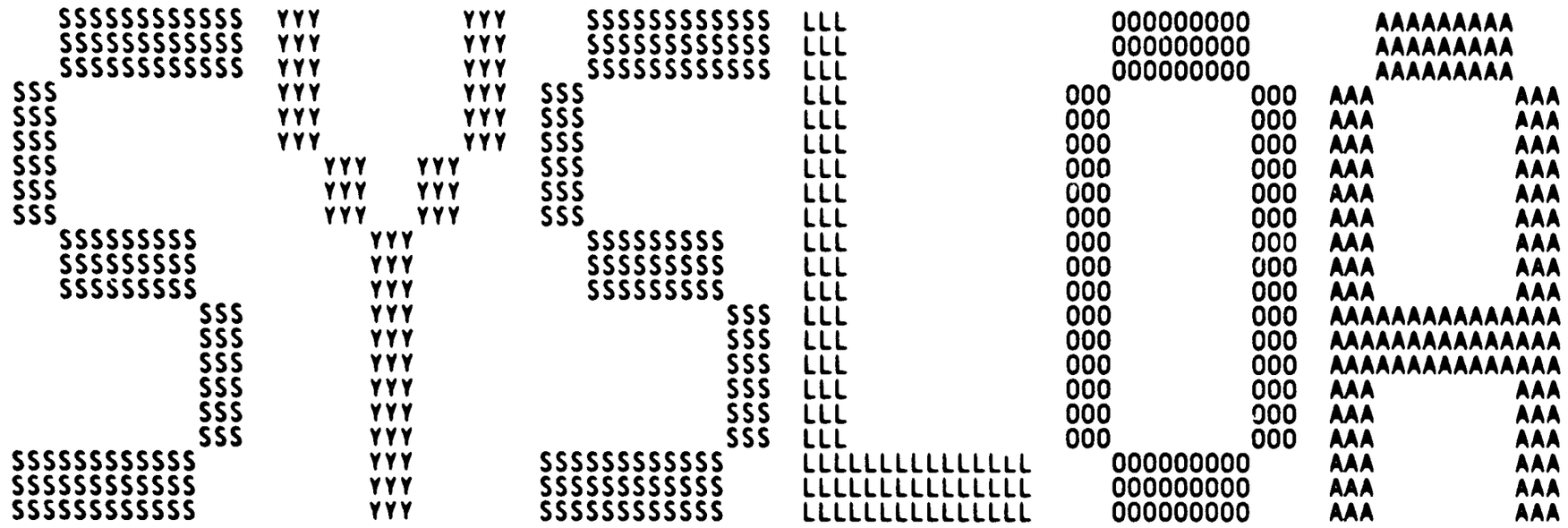
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSSSSSSSSSS YYY YYY SSSSSSSSSSS LLL 000000000 AAAAAAAAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSS YYY YYY SSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSSSSSSSS YYY SSSSSSSSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAAAAAAAAAAAAAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSS YYY SSS LLL 000 000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA
SSSSSSSSSSS YYY SSSSSSSSSSS LLLLLLLLLLLLLLL 000000000 AAA AAA

```
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     1          .TITLE  FALMAN - Cluster Failover Manager
0000     2          .IDENT  'V04-000'
0000     3 ;
0000     4 ;****************************************************************************
0000     5 ;*                                                                          *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     8 ;*  ALL RIGHTS RESERVED.                                                    *
0000     9 ;*                                                                          *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    15 ;*  TRANSFERRED.                                                            *
0000    16 ;*                                                                          *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT  *
0000    19 ;*  CORPORATION.                                                            *
0000    20 ;*                                                                          *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    23 ;*                                                                          *
0000    24 ;*                                                                          *
0000    25 ;****************************************************************************
0000    26 ;
0000    27
0000    28 ;++
0000    29 ; FACILITY: EXECUTIVE, CLUSTER MANAGEMENT
0000    30 ;
0000    31 ; ABSTRACT:
0000    32 ;       This module contains the routines that direct failover in a
0000    33 ;       VAXcluster.
0000    34 ;
0000    35 ; ENVIRONMENT: VAX/VMS
0000    36 ;
0000    37 ; AUTHOR: David W. Thiel,        CREATION DATE: 24-May-1983
0000    38 ;
0000    39 ; MODIFIED BY:
0000    40 ;
0000    41 ;       V03-014 DWT0225         David W. Thiel          11-Jul-1984
0000    42 ;               Change call to temporary name EXE$MNTVERSP2 to
0000    43 ;               real name EXE$CLUTRANIO.
0000    44 ;
0000    45 ;       V03-013 DWT0222         David W. Thiel          25-Jun-1984
0000    46 ;               Revise CNX$FAILO_SYNC to minimize time delays
0000    47 ;               in synchronizing failover steps.
0000    48 ;
0000    49 ;       V03-012 DWT0220         David W. Thiel          8-May-1984
0000    50 ;               Correct previous fix to handle reference to symbol
0000    51 ;               outside of this image.
0000    52 ;
0000    53 ;       V03-011 DWT0219         David W. Thiel          8-May-1984
0000    54 ;               Add synchronization between I/O and lock manager
0000    55 ;               clusters.
0000    56 ;
0000    57 ;       V03-010 DWT0188         David W. Thiel          9-Mar-1984
```

```
0000     58 ;              Correct synchronization routine to correctly drop
0000     59 ;              thread when it is discovered that another failover
0000     60 ;              has begun.
0000     61 ;
0000     62 ;       V03-009 SRB0110         Steve Beckhardt         2-Feb-1984
0000     63 ;              Added more failover table entries for improved
0000     64 ;              consistency checking.
0000     65 ;
0000     66 ;       V03-008 DWT0154         David W. Thiel          29-Dec-1983
0000     67 ;              Use single table for addition and removal of nodes.
0000     68 ;              Change names of CNX$BEGIN_FAILIN and
0000     69 ;              CNX$BEGIN_FAILOVER to CNX$MEMBERSHIP_CHANGE.
0000     70 ;              Remove obsolete failover table entries and
0000     71 ;              CNX$FAILO_QWAIT routine.
0000     72 ;
0000     73 ;       V03-007 DWT0137         David W  Thiel          07-Oct-1983
0000     74 ;              Differentiate cases of a certain future failover
0000     75 ;              (based on existence of a broken connection to a
0000     76 ;              cluster member) and a pending failover (node has
0000     77 ;              been failed out, but failover table processing
0000     78 ;              has not begun due to the state of a previous
0000     79 ;              instance of failover table processing) using the
0000     80 ;              CLUB$V_LOST_CNX bit.
0000     81 ;
0000     82 ;       V03-006 DWT0121         David W. Thiel          20-Aug-1983
0000     83 ;              Add failover table entry to start journal recovery.
0000     84 ;
0000     85 ;       V03-005 DWT0116         David W. Thiel          1-Aug-1983
0000     86 ;              Correct synchronization logic.  Correct quorum
0000     87 ;              waiting after a failover.
0000     88 ;
0000     89 ;       V03-004 DWT0111         David W. Thiel          27-July-1983
0000     90 ;              Add fail-in table.
0000     91 ;              Convert interface to failover routines from JMP to
0000     92 ;              JSB.
0000     93 ;
0000     94 ;       V03-003 SRB0095         Steve Beckhardt         9-Jun-1983
0000     95 ;              Add entries to failover table.
0000     96 ;
0000     97 ;       V03-002 DWT0104         David W. Thiel          8-June-1983
0000     98 ;              Add failover routine CNX$FAILO_QWAIT that waits for
0000     99 ;              quorum before completing.
0000    100 ;              Add CNX$CHECK_FAILOVER to test for and initiate
0000    101 ;              a pending failover.
0000    102 ;
0000    103 ;       V03-001 DWT0103         David W. Thiel          27-May-1983
0000    104 ;              Add index definition argument to FSTEP macro.
0000    105 ;              Add jacket routines for calling existing failover
0000    106 ;              routines.
0000    107 ;              Avoid trying to send message to a local node.
0000    108 ;              Fix logic error in synchronization code.
0000    109 ;
0000    110 ;--
0000    111
```

```
                                C 15
0000    113          .SBTTL  DECLARATIONS
0000    114 ;
0000    115 ; INCLUDE FILES:
0000    116 ;
0000    117          $CDRPDEF                        ; CDRP offsets
0000    118          $CLSMSGDEF                      ; Cluster message definitions
0000    119          $CLUBDEF                        ; CLUster Block offsets
0000    120          $CSBDEF                         ; CSB Offsets
0000    121          $FKBDEF                         ; Fork block offsets
0000    122          $IPLDEF                         ; IPL definitions
0000    123          $SSDEF                          ; Status code definitions
0000    124
0000    125 ;
0000    126 ; MACRO DEFINITIONS:
0000    127 ;
0000    128
0000    129          .MACRO  FSTEP   ADDR,STEP_INDEX,?LABEL
0000    130          .SHOW   BINARY
0000    131                  ; Self relative address
0000    132                  LABEL:  .LONG   ADDR-LABEL
0000    133          .IF NB  STEP_INDEX
0000    134          .SHOW   EXPANSIONS
0000    135                  STEP_INDEX=     <LABEL-CNX$FAILOVER_TABLE>@<-2>
0000    136          .NOSHOW EXPANSIONS
0000    137          .ENDC
0000    138          .NOSHOW BINARY
0000    139          .ENDM   FSTEP
0000    140
0000    141 ;****************** T E M P O R A R Y   D E F I N I T I O N S : *********
0000    142 ;
0000    143 .IIF NOT_DEFINED CLUFCB$V_WAITING,      CLUFCB$V_WAITING=       CLUFCB$V_FKB_BUSY+1
0000    144 .IIF NOT_DEFINED CLUFCB$M_WAITING,      CLUFCB$M_WAITING=       1@CLUFCB$V_WAITING
0000    145 ;
0000    146 ;*******************************************************************************
0000    147
0000    148
0000    149 ;*******************************************************************************
0000    150 ;
0000    151 ; NOTE: The following assumptions are in effect for this entire module.
0000    152 ;
0000    153 ;*******************************************************************************
0000    154
0000    155          ASSUME  IPL$_SYNCH  EQ  IPL$_SCS
0000    156          ASSUME  IPL$_SYNCH  EQ  IPL$_TIMER
```

```
0000    158          .SBTTL  CNX$FAILOVER_TABLE - Failover Sequencing Table
0000    159
0000    160 ;++
0000    161 ;
0000    162 ; FUNCTIONAL DESCRIPTION:
0000    163 ;
0000    164 ;       The failover table lists the sequence of steps in failing nodes
0000    165 ;       out of a cluster.  Each table entry is the self-relative address
0000    166 ;       of a routine that performs one step.
0000    167 ;
0000    168 ;       If any failover step breaks the thread of IPL SYNC execution, it
0000    169 ;       is possible that another failover may be needed at that point.
0000    170 ;       The way in which this is handled depends on a mode of failover
0000    171 ;       table processing.  In NOREFAIL mode, handling of a subsequent
0000    172 ;       failover until either this mode is left or the end of table
0000    173 ;       processing is reached.  In REFAIL mode, this failover is
0000    174 ;       abandoned and processing of the table entries is reinitiated
0000    175 ;       from the beginning.
0000    176 ;
0000    177 ;       The table is processed as follows.  Each node independently
0000    178 ;       executes the steps in the failover table, synchronized only at
0000    179 ;       the begining and when specific requests for synchronization
0000    180 ;       are made.  Control of synchronization and other aspects of
0000    181 ;       failover processing are themselves controlled by table entries.
0000    182 ;       These special tables entries are described next.
0000    183 ;
0000    184 ;    CNX$FAILO_NOP -- No Operation
0000    185 ;
0000    186 ;       NOP entry.  Used to reserve space for patching.
0000    187 ;
0000    188 ;    CNX$FAILO_END -- End of Table
0000    189 ;
0000    190 ;       Marks the end of the table and terminates table processing.
0000    191 ;
0000    192 ;    CNX$FAILO_REFAIL -- Set Re-failoverable Mode
0000    193 ;
0000    194 ;       Specifies that this failover may be abandoned at any time and
0000    195 ;       another begun.  In this mode, any suspended thread must save
0000    196 ;       the failover identification and validate it upon being
0000    197 ;       resumed.  If the ID has changed, the thread must be terminated;
0000    198 ;       otherwise, it may proceed.
0000    199 ;
0000    200 ;    CNX$FAILO_NOREFAIL -- Clear Re-failoverable Mode
0000    201 ;
0000    202 ;       Specifies that this failover may not be abandoned and that
0000    203 ;       no new failover may be begun.  While in this mode, failover
0000    204 ;       routines MAY NOT suspend themselves without a guarantee of
0000    205 ;       resumption.  In particular, a routine MAY NOT wait for a
0000    206 ;       message to arrive from the corresponding routine on another
0000    207 ;       node.  Furthermore, a routine MAY NOT wait for process
0000    208 ;       execution.
0000    209 ;
0000    210 ;    CNX$FAILO_SYNC -- Synchronize Failover Steps
0000    211 ;
0000    212 ;       Requests synchronization at this step.  One node is selected
0000    213 ;       as the synchronizer when failover table processing is
0000    214 ;       initiated.  When this table entry is encountered, every
```

```
0000    215 ;       node sends a READY message to the synchronizing node.  When
0000    216 ;       the synchronizing node executes this entry, it waits for a
0000    217 ;       READY from every other node involved in the failover and
0000    218 ;       then sends a DOSTEP message to every involved node.  It
0000    219 ;       and the other nodes then complete this step and proceed to
0000    220 ;       the next.
0000    221 ;
0000    222 ;       Following a synchronization step, it is guaranteed that every
0000    223 ;       other involved node has completed the step preceding the
0000    224 ;       synchronization step.  Due to delays and queuing of messages,
0000    225 ;       it is still possible to receive a message from another node
0000    226 ;       that was queued before the synchronization step.
0000    227 ;
0000    228 ;       Requesting a synchronization step ALWAYS puts the failover
0000    229 ;       into Re-Failoverable Mode.  This results in the very substantial
0000    230 ;       simplification which allows any communications problems during
0000    231 ;       synchronization to be handled as part of a subsequent failover.
0000    232 ;
0000    233 ; The following description applies to every failover routine:
0000    234 ;
0000    235 ; CALLING SEQUENCE:
0000    236 ;
0000    237 ;       Invoked by:
0000    238 ;               JSB failover_routine
0000    239 ;       IPL is IPL$_SYNC = IPL$_SCS = IPL$_TIMER
0000    240 ;       To continue failover processing, return with:
0000    241 ;               RSB
0000    242 ;       To abandon failover processing in REFAIL mode:
0000    243 ;               ADDL    #4,SP
0000    244 ;               RSB
0000    245 ;       To terminate failover processing in either mode:
0000    246 ;               ADDL    #4,SP
0000    247 ;               JMP     CNX$END_FAILOVER
0000    248 ;
0000    249 ;       A failover routine runs as a fork process and must
0000    250 ;       behave according to the general rules for fork processes.
0000    251 ;
0000    252 ; INPUT PARAMETERS:
0000    253 ;
0000    254 ;       04(SP): Address of caller's caller -- return here if thread
0000    255 ;                   is suspended
0000    256 ;       00(SP): Return here to continue failover processing
0000    257 ;
0000    258 ;       R5:     Address of CLUster Failover Control Block (CLUFCB)
0000    259 ;       R4:     Address of CLUster Block (CLUB)
0000    260 ;       R3:     Failover ID (copy of CLUFCB$L_ID(R5))
0000    261 ;
0000    262 ; OUTPUT PARAMETERS:
0000    263 ;
0000    264 ;       NONE
0000    265 ;
0000    266 ; COMPLETION CODES:
0000    267 ;
0000    268 ;       NONE
0000    269 ;
0000    270 ; SIDE EFFECTS:
0000    271 ;
```

```
                     0000   272 ;       Registers R0-R5 need not be preserved.  Everything interesting
                     0000   273 ;       that a failover does is a side effect of some sort.
                     0000   274 ;
                     0000   275 ;--
                     0000   276
                 00000000   277         .PSECT  $$$060,LONG                     ; Read only Data PSECT
                     0000   278
                     0000   279 CNX$FAILOVER_TABLE::
                     0000   280         FSTEP   CNX$FAILO_NOREFAIL              ; Must be first entry
000000A4'            0000                 30000$: .LONG   CNX$FAILO_NOREFAIL-30000$
                     0004   281
                     0004   282         FSTEP   LCK$STALL_ALL                   ; Stall lock requests
FFFFFFFC'            0004                 30001$: .LONG   LCK$STALL_ALL-30001$
                     0008   283 ;
                     0008   284 ; Everything up to this point must be synchronous with the beginning
                     0008   285 ; of failover table execution.  Note that the beginning of failover
                     0008   286 ; table execution is not necessarily synchronous with the receipt of
                     0008   287 ; the phase 2 message in the case when a failover is already in
                     0008   288 ; progress.
                     0008   289 ;
                     0008   290         FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000097'            0008                 30002$: .LONG   CNX$FAILO_REFAIL-30002$
                     000C   291         FSTEP   CNX$FAILO_SYNC                  ; Synchronize
000000B5'            000C                 30003$: .LONG   CNX$FAILO_SYNC-30003$
                     0010   292         FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000094'            0010                 30004$: .LONG   CNX$FAILO_NOREFAIL-30004$
                     0014   293
                     0014   294         FSTEP   CNX$IO_SYNCH                    ; Serialize I/O
000000A7'            0014                 30005$: .LONG   CNX$IO_SYNCH-30005$
                     0018   295
                     0018   296         FSTEP   LCK$SET_STATE1                  ; Set rebuild state 1
FFFFFFE8'            0018                 30006$: .LONG   LCK$SET_STATE1-30006$
                     001C   297
                     001C   298         FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000083'            001C                 30007$: .LONG   CNX$FAILO_REFAIL-30007$
                     0020   299         FSTEP   CNX$FAILO_SYNC                  ; Synchronize
000000A1'            0020                 30008$: .LONG   CNX$FAILO_SYNC-30008$
                     0024   300         FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000080'            0024                 30009$: .LONG   CNX$FAILO_NOREFAIL-30009$
                     0028   301
                     0028   302         FSTEP   LCK$INIT_REBUILD                ; Remove all master copy locks
FFFFFFD8'            0028                 30010$: .LONG   LCK$INIT_REBUILD-30010$
                     002C   303                                                 ;  and directory entries
                     002C   304
                     002C   305         FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000073'            002C                 30011$: .LONG   CNX$FAILO_REFAIL-30011$
                     0030   306         FSTEP   CNX$FAILO_SYNC                  ; Synchronize
00000091'            0030                 30012$: .LONG   CNX$FAILO_SYNC-30012$
                     0034   307         FSTEP   CNX$FAILO_NOREFAIL              ; Don't allow re-failovers
00000070'            0034                 30013$: .LONG   CNX$FAILO_NOREFAIL-30013$
                     0038   308
                     0038   309         FSTEP   LCK$SET_STATE2                  ; Set rebuild state 2
FFFFFFC8'            0038                 30014$: .LONG   LCK$SET_STATE2-30014$
                     003C   310
                     003C   311         FSTEP   CNX$FAILO_REFAIL                ; Must precede synchronize
00000063'            003C                 30015$: .LONG   CNX$FAILO_REFAIL-30015$
                     0040   312         FSTEP   CNX$FAILO_SYNC                  ; Synchronize
```

```
00000081' 0040                          30016$: .LONG   CNX$FAILO_SYNC-30016$
          0044   313          FSTEP    CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000060' 0044                          30017$: .LONG   CNX$FAILO_NOREFAIL-30017$
          0048   314
          0048   315          FSTEP    LCK$REBUILD_LKBS        ; Rebuild locks
FFFFFFB8' 0048                          30018$: .LONG   LCK$REBUILD_LKBS-30018$
          004C   316
          004C   317          FSTEP    CNX$FAILO_REFAIL        ; Must precede synchronize
00000053' 004C                          30019$: .LONG   CNX$FAILO_REFAIL-30019$
          0050   318          FSTEP    CNX$FAILO_SYNC          ; Synchronize
00000071' 0050                          30020$: .LONG   CNX$FAILO_SYNC-30020$
          0054   319          FSTEP    CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000050' 0054                          30021$: .LONG   CNX$FAILO_NOREFAIL-30021$
          0058   320
          0058   321          FSTEP    LCK$SET_STATE3          ; Set rebuild state 3
FFFFFFA8' 0058                          30022$: .LONG   LCK$SET_STATE3-30022$
          005C   322
          005C   323          FSTEP    CNX$FAILO_REFAIL        ; Must precede synchronize
00000043' 005C                          30023$: .LONG   CNX$FAILO_REFAIL-30023$
          0060   324          FSTEP    CNX$FAILO_SYNC          ; Synchronize
00000061' 0060                          30024$: .LONG   CNX$FAILO_SYNC-30024$
          0064   325          FSTEP    CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000040' 0064                          30025$: .LONG   CNX$FAILO_NOREFAIL-30025$
          0068   326
          0068   327          FSTEP    LCK$REBUILD_RSBS        ; Rebuild resources, grant
FFFFFF98' 0068                          30026$: .LONG   LCK$REBUILD_RSBS-30026$
          006C   328                                    ; unprotected locks
          006C   329
          006C   330          FSTEP    CNX$FAILO_REFAIL        ; Must precede synchronize
00000033' 006C                          30027$: .LONG   CNX$FAILO_REFAIL-30027$
          0070   331          FSTEP    CNX$FAILO_SYNC          ; Synchronize
00000051' 0070                          30028$: .LONG   CNX$FAILO_SYNC-30028$
          0074   332          FSTEP    CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000030' 0074                          30029$: .LONG   CNX$FAILO_NOREFAIL-30029$
          0078   333
          0078   334          FSTEP    LCK$SET_STATE0          ; Set rebuild state 0
FFFFFF88' 0078                          30030$: .LONG   LCK$SET_STATE0-30030$
          007C   335
          007C   336          FSTEP    CNX$FAILO_REFAIL        ; Must precede synchronize
00000023' 007C                          30031$: .LONG   CNX$FAILO_REFAIL-30031$
          0080   337          FSTEP    CNX$FAILO_SYNC          ; Synchronize
00000041' 0080                          30032$: .LONG   CNX$FAILO_SYNC-30032$
          0084   338
          0084   339          FSTEP    LCK$RESUME_UNPROT       ; Resume processes waiting for
FFFFFF7C' 0084                          30033$: .LONG   LCK$RESUME_UNPROT-30033$
          0088   340                                    ; unprotected locks
          0088   341
          0088   342          FSTEP    CNX$FAILO_JNL           ; CSP/RCP recovery initiation
00000223' 0088                          30034$: .LONG   CNX$FAILO_JNL-30034$
          008C   343
          008C   344          FSTEP    CNX$FAILO_SYNC          ; Synchronize
00000035' 008C                          30035$: .LONG   CNX$FAILO_SYNC-30035$
          0090   345
          0090   346          FSTEP    LCK$RESUME_ALL          ; Resume processes waiting for any
FFFFFF70' 0090                          30036$: .LONG   LCK$RESUME_ALL-30036$
          0094   347                                    ;  type of lock request
          0094   348
```

```
                0094    349         FSTEP   CNX$FAILO_NOREFAIL      ; Don't allow re-failovers
00000010'       0094                        30037$: .LONG   CNX$FAILO_NOREFAIL-30037$
                0098    350
                0098    351         FSTEP   LCK$REBUILD_RSBS         ; Rebuild resources, grant
FFFFFF68'       0098                        30038$: .LONG   LCK$REBUILD_RSBS-30038$
                009C    352                                         ;  protected (all) locks
                009C    353
                009C    354         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    355         FSTEP   CNX$FAILO_END              ; End of table
00000017'       00A0                        30040$: .LONG   CNX$FAILO_END-30040$
                00A4    356 ;
000000C4        00A4    357         .BLKL   8                           ; Reserve patch space
                00C4    358 ;
                00C4    359 ;
          00000000      360         .PSECT  $$$100,LONG
                0000    361
                0000    362         .DEFAULT        DISPLACEMENT,WORD
```

```
                                0000   364          .SBTTL  CNX$MEMBERSHIP_CHANGE - Begin membership change actions
                                0000   365 ;++
                                0000   366 ;
                                0000   367 ; FUNCTIONAL DESCRIPTION:
                                0000   368 ;
                                0000   369 ;  CNX$MEMBERSHIP_CHANGE
                                0000   370 ;       This routine is called whenever a node or nodes are added to
                                0000   371 ;       or removed from a cluster, including initial cluster formation.
                                0000   372 ;       Actions are undertaking by the members of the cluster to
                                0000   373 ;       adjust to the new cluster membership.  If an uninterruptible
                                0000   374 ;       action is already in progress, handling of the new event is deferred.
                                0000   375 ;
                                0000   376 ; CALLING SEQUENCE:
                                0000   377 ;
                                0000   378 ;       JSB     CNX$MEMBERSHIP_CHANGE
                                0000   379 ;       IPL is IPL$_SCS
                                0000   380 ;
                                0000   381 ; INPUT PARAMETERS:
                                0000   382 ;
                                0000   383 ;       NONE
                                0000   384 ;
                                0000   385 ; OUTPUT PARAMETERS:
                                0000   386 ;
                                0000   387 ;       NONE
                                0000   388 ;
                                0000   389 ; COMPLETION CODES:
                                0000   390 ;
                                0000   391 ;       NONE
                                0000   392 ;
                                0000   393 ; SIDE EFFECTS:
                                0000   394 ;
                                0000   395 ;       R0-R5 are destroyed.
                                0000   396 ;
                                0000   397 ;--
                                0000   398
                                0000   399          .ENABLE LSB
                                0000   400 CNX$MEMBERSHIP_CHANGE::
54   00000000'GF    D0          0000   401 1$:      MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
     55    010C C4  9E          0007   402          MOVAB   CLUB$B_CLUFCB(R4),R5    ; Address of failover control block
  05 20 A5    00  E1          000C   403          BBC     #CLUFCB$V_ACTIVE, -     ; Branch if no active failover and
                                0011   404                  CLUFCB$L_STATUS(R5),10$ ;   set active failover flag
     20 A5    02  C8          0011   405          BISL2   #CLUFCB$M_PENDING, -    ; Set failover pending flag
                                0015   406                  CLUFCB$L_STATUS(R5)
                           05   0015   407          RSB                             ; Return, will do failover later
                                0016   408
     20 A5    06  CA          0016   409 10$:     BICL2   #<CLUFCB$M_PENDING ! -  ; Clear failover pending flag
                                001A   410                    CLUFCB$M_SYNC_NODE>, - ;  and local synchronizing node flags
                                001A   411                  CLUFCB$L_STATUS(R5)
  1C A5    34 A4    D0          001A   412          MOVL    CLUB$L_LST_XTN(R4), -   ; Make last transition ID the failover
                                001F   413                  CLUFCB$L_ID(R5)         ;   identifier
           18 A5    D4          001F   414          CLRL    CLUFCB$L_STEP(R5)       ; Initialize failover step
  24 A5    5C A4    D0          0022   415          MOVL    CLUB$L_COORD(R4), -     ; Set up CSB address of synchronizer node
                                0027   416                  CLUFCB$L_SYNC_CSB(R5)
  10 A4    5C A4    D1          0027   417          CMPL    CLUB$L_COORD(R4), -     ; Was this node the failover coordinator?
                                002C   418                  CLUB$L_LOCAL_CSB(R4)
              04    12          002C   419          BNEQ    20$                     ; Branch if no
     20 A5    04    C8          002E   420          BISL2   #CLUFCB$M_SYNC_NODE, -  ; Make this node the failover sync node
```

```
                                0032   421                      CLUFCB$L_STATUS(R5)
                          55    DD    0032   422 20$:    PUSHL   R5                      ; Save CLUFCB address
                                0034   423                ASSUME  <CLUFCB$B_NODEMAP+CLUFCB$S_NODEMAP> EQ CLUFCB$B_RESPMAP
28 A5   0040 8F   00    6E    00    2C    0034   424                MOVC5   #0,(SP),#0, -            ; Clear node map and response map
                                0038   425                        #CLUFCB$S_NODEMAP+CLUFCB$S_RESPMAP, -
                                003D   426                        CLUFCB$B_NODEMAP(R5)
                          20    BA    003D   427                POPR    #^M<R5>                 ; Restore CLUFCB address
                      FFBE'     30    003F   428                BSBW    CNX$SCAN_CSBS           ; Iterate over all CSBs
                     24 50      E9    0042   429                BLBC    R0,40$                  ; Branch when done
                0E 60 A3        01    E1    0045   430                BBC     #CSB$V_MEMBER, -        ; Branch if CSB is not for a member node
                                004A   431                        CSB$L_STATUS(R3),30$
                09 60 A3        00    E0    004A   432                BBS     #CSB$V_LONG_BREAK, -    ; Branch if connection permanently broken
                                004F   433                        CSB$L_STATUS(R3),30$
                   50    4C A3   3C    004F   434                MOVZWL  CSB$W_CSID_IDX(R3),R0   ; CSID index of member node
                00 28 A5    50    E3    0053   435                BBCS    R0, -                   ; Mark node present
                                0058   436                        CLUFCB$B_NODEMAP(R5),30$
                                05    0058   437 30$:    RSB
                                0059   438
                                0059   439 ;
                                0059   440 ; Failover routine may jump here to exit failover processing
                                0059   441 ;
                                0059   442 CNX$END_FAILOVER::
             54   00000000'GF    D0    0059   443                MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
                55    010C C4    9E    0060   444                MOVAB   CLUB$B_CLUFCB(R4),R5    ; Address of failover control block
                  18 A5    27    D0    0065   445                MOVL    #FORCE_END, -           ; Set index to force end of table processing
                                0069   446                        CLUFCB$L_STEP(R5)
                                0069   447 ;
                                0069   448 ; Begin a failover step
                                0069   449 ;
                   53    1C A5    D0    0069   450 40$:    MOVL    CLUFCB$L_ID(R5),R3      ; Failover ID
                   50    18 A5    D0    006D   451                MOVL    CLUFCB$L_STEP(R5),R0    ; Get index of next failover step
             50    0000'CF40     DE    0071   452                MOVAL   W^CNX$FAILOVER_TABLE[R0],R0 ; Get table entry
                                0077   453 ;
                                0077   454 ;       R5:     Address of failover control block (with available fork block at head
                                0077   455 ;       R4:     Address of CLUB
                                0077   456 ;       R3:     Failover ID
                                0077   457 ;
                      00 B040    16    0077   458                JSB     @(R0)[R0]               ; Convert self relative to absolute address
             54   00000000'GF    D0    007B   459                MOVL    G^CLU$GL_CLUB,R4        ; Address of CLUB
                55    010C C4    9E    0082   460                MOVAB   CLUB$B_CLUFCB(R4),R5    ; Address of failover control block
                                0087   461 ;
                                0087   462 ; Use the bits CLUFCB$V_ACTIVE, CLUFDB$V_PENDING, and CLUB$V_LOST_CNX to decide whet
                                0087   463 ; to continue this failover, start again at the beginning of the table, or drop ever
                                0087   464 ; knowing that a new failover will begin soon (the last case is assured if a connect
                                0087   465 ; to a cluster member is broken).
                                0087   466 ;
                0A 20 A5        00    E0    0087   467                BBS     #CLUFCB$V_ACTIVE, -     ; Branch if new failover may not
                                008C   468                        CLUFCB$L_STATUS(R5),50$ ;  commence here
                0A 20 A5        01    E0    008C   469                BBS     #CLUFCB$V_PENDING, -    ; Branch if another failover is pending
                                0091   470                        CLUFCB$L_STATUS(R5),60$ ;  and initiate it
                08 1C A4        17    E0    0091   471                BBS     #CLUB$V_LOST_CNX, -     ; Branch if a connection has been lost and
                                0096   472                        CLUB$L_FLAGS(R4),70$    ;  and new failover will soon happen
                      18 A5     D6    0096   473 50$:    INCL    CLUFCB$L_STEP(R5)       ; Advance to next failover routine
                         CE     11    0099   474                BRB     40$                     ; Do next step
                                009B   475
                       FF62     31    009B   476 60$:    BRW     1$                      ; Start a new failover
                                009E   477
```

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

```
                     009F    482          .SBTTL  CNX$FAILO_REFAIL - Allow new failovers to start
                     009F    483
                     009F    484 ;++
                     009F    485 ;
                     009F    486 ; FUNCTIONAL DESCRIPTION:
                     009F    487 ;
                     009F    488 ;       Enter REFAIL mode in which new executions of the failover
                     009F    489 ;       table may be initiated.
                     009F    490 ;
                     009F    491 ; CALLING SEQUENCE:
                     009F    492 ;
                     009F    493 ;       JSB     CNX$FAILO_REFAIL
                     009F    494 ;       IPL is IPL$_SCS
                     009F    495 ;
                     009F    496 ; INPUT PARAMETERS:
                     009F    497 ;
                     009F    498 ;       R5:     Address of Failover Control Block
                     009F    499 ;       R4:     Address of CLUB
                     009F    500 ;       R3:     Failover ID
                     009F    501 ;
                     009F    502 ; OUTPUT PARAMETERS:
                     009F    503 ;
                     009F    504 ;       NONE
                     009F    505 ;
                     009F    506 ; COMPLETION CODES:
                     009F    507 ;
                     009F    508 ;       NONE
                     009F    509 ;
                     009F    510 ; SIDE EFFECTS:
                     009F    511 ;
                     009F    512 ;       NONE
                     009F    513 ;
                     009F    514 ;--
                     009F    515
                     009F    516 CNX$FAILO_REFAIL::
20 A5    01    CA    009F    517          BICL2   #CLUFCB$M_ACTIVE, -     ; Clear active bit
                     00A3    518                  CLUFCB$L_STATUS(R5)
                  05 00A3    519          RSB
```

```
                        00A4    521          .SBTTL  CNX$FAILO_NOREFAIL - Prevent second entrance to failover table
                        00A4    522
                        00A4    523 ;++
                        00A4    524 ;
                        00A4    525 ; FUNCTIONAL DESCRIPTION:
                        00A4    526 ;
                        00A4    527 ;
                        00A4    528 ;       Stop allowing new failovers to begin.
                        00A4    529 ;       Terminate any previous failover that may have been in progress.
                        00A4    530 ;
                        00A4    531 ; CALLING SEQUENCE:
                        00A4    532 ;
                        00A4    533 ;       JSB     CNX$FAILO_NOREFAIL
                        00A4    534 ;       IPL is IPL$_SCS
                        00A4    535 ;
                        00A4    536 ; INPUT PARAMETERS:
                        00A4    537 ;
                        00A4    538 ;       R5:     Address of  ailover Control Block
                        00A4    539 ;       R4:     Address of CLJB
                        00A4    540 ;       R3:     Failover ID
                        00A4    541 ;
                        00A4    542 ; OUTPUT PARAMETERS:
                        00A4    543 ;
                        00A4    544 ;       NONE
                        00A4    545 ;
                        00A4    546 ; COMPLETION CODES:
                        00A4    547 ;
                        00A4    548 ;       NONE
                        00A4    549 ;
                        00A4    550 ; SIDE EFFECTS:
                        00A4    551 ;
                        00A4    552 ;       NONE
                        00A4    553 ;
                        00A4    554 ;--
                        00A4    555
                        00A4    556 CNX$FAILO_NOREFAIL::
0C 20 A5    00    E2    00A4    557          BBSS    #CLUFCB$V_ACTIVE, -     ; Set active bit and branch
                        00A9    558                  CLUFCB$L_STATUS(R5),10$ ;   if it was already set
07 20 A5    03    E5    00A9    559          BBCC    #CLUFCB$V_FKB_BUSY, -   ; Branch if fork block free
                        00AE    560                  CLUFCB$L_STATUS(R5),10$ ;   and mark it free
      50    65    0F    00AE    561          REMQUE  (R5),R0                 ; Remove from queue
   20 A5    10    CA    00B1    562          BICL2   #CLUFCB$M_WAITING, -    ; Clear waiting bit
                        00B5    563                  CLUFCB$L_STATUS(R5)
                  05    00B5    564 10$:     RSB
```

```
                        .SBTTL  CNX$FAILO_NOP - Null failover routine

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

CNX$FAILO_NOP::
05              RSB
```

```
                        00B7   603          .SBTTL  CNX$FAILO_END - End failover table processing
                        00B7   604
                        00B7   605 ;++
                        00B7   606 ;
                        00B7   607 ; FUNCTIONAL DESCRIPTION:
                        00B7   608 ;
                        00B7   609 ;       End failover table processing.  This assumes that REFAIL
                        00B7   610 ;       mode is in effect.
                        00B7   611 ;
                        00B7   612 ; CALLING SEQUENCE:
                        00B7   613 ;
                        00B7   614 ;       JSB     CNX$FAILO_END
                        00B7   615 ;       IPL is IPL$_SCS
                        00B7   616 ;
                        00B7   617 ; INPUT PARAMETERS:
                        00B7   618 ;
                        00B7   619 ;       R5:     Address of Failover Control Block
                        00B7   620 ;       R4:     Address of CLUB
                        00B7   621 ;       R3:     Failover ID
                        00B7   622 ;
                        00B7   623 ; OUTPUT PARAMETERS:
                        00B7   624 ;
                        00B7   625 ;       NONE
                        00B7   626 ;
                        00B7   627 ; COMPLETION CODES:
                        00B7   628 ;
                        00B7   629 ;       NONE
                        00B7   630 ;
                        00B7   631 ; SIDE EFFECTS:
                        00B7   632 ;
                        00B7   633 ;       NONE
                        00B7   634 ;
                        00B7   635 ;--
                        00B7   636
                        00B7   637 CNX$FAILO_END::
5E   04   C0            00B7   638          ADDL2   #4,SP                   ; Remove caller's address
                  05    00BA   639          RSB                             ; Return to caller's caller
                        00BB   640
```

```
                        00BB   642          .SBTTL  CNX$IO_SYNCH - Synchronize with I/O cluster
                        00BB   643
                        00BB   644 ;++
                        00BB   645 ;
                        00BB   646 ; FUNCTIONAL DESCRIPTION:
                        00BB   647 ;
                        00BB   648 ;       Synchronize with the I/O cluster.
                        00BB   649 ;
                        00BB   650 ;       Serialize I/O to multi-host disks to ensure that I/O issued under current lo
                        00BB   651 ;       completed before I/O issued under locks granted after some locks have been r
                        00BB   652 ;       as the result of removing nodes from the cluster.
                        00BB   653 ;
                        00BB   654 ;       Note that it is unnecessary to do this operation if nodes are being added an
                        00BB   655 ;       are being removed.
                        00BB   656 ;
                        00BB   657 ;       This entry is called following a synchronization call to ensure that all nod
                        00BB   658 ;       have disconnected from a node being removed before this call is made.  It is
                        00BB   659 ;       important that the node being removed has either lost quorum or crashed and
                        00BB   660 ;       therefore ceased to issue I/O's before this call is made.
                        00BB   661 ;
                        00BB   662 ; CALLING SEQUENCE:
                        00BB   663 ;
                        00BB   664 ;       JSB     CNX$IO_SYNCH
                        00BB   665 ;       IPL is IPL$_SCS
                        00BB   666 ;
                        00BB   667 ; INPUT PARAMETERS:
                        00BB   668 ;
                        00BB   669 ;       R5:     Address of Failover Control Block
                        00BB   670 ;       R4:     Address of CLUB
                        00BB   671 ;       R3:     Failover ID
                        00BB   672 ;
                        00BB   673 ; OUTPUT PARAMETERS:
                        00BB   674 ;
                        00BB   675 ;       NONE
                        00BB   676 ;
                        00BB   677 ; COMPLETION CODES:
                        00BB   678 ;
                        00BB   679 ;       NONE
                        00BB   680 ;
                        00BB   681 ; SIDE EFFECTS:
                        00BB   682 ;
                        00BB   683 ;       Disks are thrown into mount verification.
                        00BB   684 ;
                        00BB   685 ;--
                        00BB   686
                        00BB   687 CNX$IO_SYNCH:
                C0000000'GF  17 00BB   688          JMP     G^EXE$CLUTRANIO         ; Synchronize with I/O cluster and return
                        00C1   689
```

```
                     00C1   691          .SBTTL  CNX$FAILO_SYNC - Inter-Node Synchronization Failover Routine
                     00C1   692
                     00C1   693 ;++
                     00C1   694 ;
                     00C1   695 ; FUNCTIONAL DESCRIPTION:
                     00C1   696 ;
                     00C1   697 ;       This routine synchronizes failover action over all involved nodes.
                     00C1   698 ;       This failover routine does not complete until all nodes have completed
                     00C1   699 ;       the previous failover step.
                     00C1   700 ;       A new failover may be begun while this routine is in progress.
                     00C1   701 ;
                     00C1   702 ;       The synchronization algorithm is a follows:
                     00C1   703 ;
                     00C1   704 ;       One node is chosen as the node to drive the synchronization.  All other
                     00C1   705 ;       nodes send a message to the sync node when this routine is executed.
                     00C1   706 ;       If the sync node NAKs the message, the node waits and tries again.  If
                     00C1   707 ;       the message fails, the node drops its thread, confident that a new
                     00C1   708 ;       failover will begin sometime and start the whole thing over.  If the
                     00C1   709 ;       message succeeds, the node then waits for a DOSTEP message, upon
                     00C1   710 ;       receipt of which it exits from this failover step.
                     00C1   711 ;
                     00C1   712 ;       Meanwhile, the sync node waits for all other nodes to report in.  When
                     00C1   713 ;       a message is received for the right failover, it is noted and ACKed.
                     00C1   714 ;       If a message is not for the current failover, it is NAKed.  After all
                     00C1   715 ;       nodes has reported, a DOSTEP message is sent to every other node.
                     00C1   716 ;       When a connection breaks the CNX$CON_BREAK routine sets a bit that
                     00C1   717 ;       eliminates that node from consideration by this routine.
                     00C1   718 ;
                     00C1   719 ; CALLING SEQUENCE:
                     00C1   720 ;
                     00C1   721 ;       JSB     CNX$FAILO_SYNC
                     00C1   722 ;       IPL is IPL$_SCS
                     00C1   723 ;
                     00C1   724 ; INPUT PARAMETERS:
                     00C1   725 ;
                     00C1   726 ;       R5:     Address of Failover Control Block
                     00C1   727 ;       R4:     Address of CLUB
                     00C1   728 ;       R3:     Failover ID
                     00C1   729 ;
                     00C1   730 ; OUTPUT PARAMETERS:
                     00C1   731 ;
                     00C1   732 ;       NONE
                     00C1   733 ;
                     00C1   734 ; COMPLETION CODES:
                     00C1   735 ;
                     00C1   736 ;       NONE
                     00C1   737 ;
                     00C1   738 ; SIDE EFFECTS:
                     00C1   739 ;
                     00C1   740 ;       NONE
                     00C1   741 ;
                     00C1   742 ;--
                     00C1   743
                     00C1   744 CNX$FAILO_SYNC::
03 20 A5   02   E0   00C1   745          BBS     #CLUFCB$V_SYNC_NODE, -  ; Branch if this is the
                     00C6   746                   CLUFCB$L_STATUS(R5), -  ;   synchronizing node
                     00C6   747                   5$
```

```
                     0098   31  00C6   748          BRW     100$                        ; Branch to code for other nodes
                            00C9   749
               50    60 A4  3C  00C9   750 5$:      MOVZWL  CLUB$W_LOCAL_CSID_IDX(R4),R0 ; Local CSID index field
            00 48 A5   50  E3  00CD   751          BBCS    R0, -                       ; Mark local node responded
                            00D2   752                  CLUFCB$B_RESPMAP(R5),7$
                       0B   11  00D2   753 7$:      BRB     14$                         ; Branch to base of loop
                            00D4   754
                  20 A5    10  C8  00D4   755 10$:     BISL2   #CLUFCB$M_WAITING, -        ; Set waiting bit
                            00D8   756                  CLUFCB$L_STATUS(R5)
                     0103   30  00D8   757          BSBW    DELAY                       ; Wait for 0-1 second
                  20 A5    10  CA  00DB   758          BICL2   #CLUFCB$M_WAITING, -        ; Clear waiting bit
                            00DF   759                  CLUFCB$L_STATUS(R5)
                     51     1F  D0  00DF   760 14$:     MOVL    #CLUFCB$S_RESPMAP-1,R1      ; Byte counter
      50    28 A541    48 A541  8B  00E2   761 15$:     BICB3   CLUFCB$B_RESPMAP(R5)[R1], - ; See if all currently involved
                            00EA   762                  CLUFCB$B_NODEMAP(R5)[R1], - ;   nodes have responded
                            00EA   763                  R0
                        E8  12  00EA   764          BNEQ    10$                         ; Found unresponded node(s)
                     F3 51  F4  00EC   765          SOBGEQ  R1,15$                      ; Iterate over all nodes
                        55  DD  00EF   766          PUSHL   R5                          ; Save register
   48 A5    20    00   6E    00  2C  00F1   767          MOVC5   #0,(SP),#0, -               ; Zero out RESPMAP for next use
                            00F8   768                  #CLUFCB$S_RESPMAP, -
                            00F8   769                  CLUFCB$B_RESPMAP(R5)
                        20  BA  00F8   770          POPR    #^M<R5>                     ; Restore register
                            00FA   771 ;
                            00FA   772 ; All reponses have been received.  Send a message to every
                            00FA   773 ; node telling it to proceed with the next step.
                            00FA   774 ;
               54    00FF  8F  3C  00FA   775          MOVZWL  #<CLUFCB$S_NODEMAP*8>-1,R4 ; Index in bitmap
                        09  11  00FF   776          BRB     30$                         ; Branch to loop entrance
                            0101   777
                            0101   778 19$:     BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
                            0105   779
                     30     BA  0105   780 20$:     POPR    #^M<R4,R5>                  ; Restore index and CLUFCB addresses
                  00D4      30  0107   781          BSBW    DELAY                       ; Wait 0-1 seconds
      04 28 A5     54     E0  010A   782 30$:     BBS     R4,CLUFCB$B_NODEMAP(R5),50$ ; Branch if in map
               F8 54     F4  010F   783 40$:     SOBGEQ  R4,30$                      ; Iterate over all possible bits
                        05  0112   784          RSB                                 ; All done, return to caller
                            0113   785
   51    00000000'GF     D0  0113   786 50$:     MOVL    G^CLU$GL_CLUSVEC,R1         ; Address of cluster vector
         53    6144      D0  011A   787          MOVL    (R1)[R4],R3                 ; Address of CSB
                  E1     18  011E   788          BGEQ    19$                         ; Not a CSB -- consistency check
      EA 60 A3     18     E0  0120   789          BBS     #CSB$V_LOCAL, -             ; Branch if local node to avoid
                            0125   790                  CSB$L_STATUS(R3),40$        ;   trying to send message to self
                     30     BB  0125   791          PUSHR   #^M<R4,R5>                  ; Save context
                  FED6'     30  0127   792          BSBW    CNX$ALLOC_CDRP_ONLY         ; Get a fork block
                  FED3'     30  012A   793          BSBW    CNX$RESOURCE_CHECK          ; Watch resource availability
                D5 50     E9  012D   794          BLBC    R0,20$                      ; Branch if no memory available
            54    04 AE   D0  0130   795          MOVL    4(SP),R4                    ; Restore CLUFCB address
            2C A5    0C   9A  0134   796          MOVZBL  #CLMCNX$K_FNC_DOSTEP, - ; Function code for message
                            0138   797                  CDRP$L_VA[1(R5)
                  0093      30  0138   798          BSBW    300$                        ; Initialize CDRP for mesage building
                  FE:2'     30  013B   799          BSBW    CNX$SEND_FORGET             ; Send message to remote node and return imm
                     30     BA  013E   800          POPR    #^M<R4,R5>                  ; Restore index and CLUFCB addresses
      BC 20 A5     03     E2  0140   801          BBSS    #CLUFCB$V_FKB_BUSY, -       ; Branch if busy and mark busy
                            0145   802                  CLUFCB$L_STATUS(R5),19$
         0B A5     08     90  0145   803          MOVB    #IPL$_SCS, -                ; Store IPL in fork block
                            0149   804                  FKB$B_FIPL(R5)
```

```
                08   BA  0149   805          POPR    #^M<R3>                         ; Save caller's address
      00000000'GF   16  014B   806          JSB     G^EXE$FORK                      ; Release control momentarily
                53   DD  0151   807          PUSHL   R3                              ; Restore caller's address
          20 A5   08   CA  0153   808          BICL2   #CLUFCB$M_FKB_BUSY, -           ; Clear busy bit
                                  0157   809                  CLUFCB$L_STATUS(R5)
    B3 20 A5    01   E1  0157   810          BBC     #CLUFCB$V_PENDING, -            ; Branch if no failover is
                                  015C   811                  CLUFCB$L_STATUS(R5),40$ ;  pending
                05   05  015C   812          RSB                                     ; Return - new failover is pending
                                  015D   813
                                  015D   814 ;
                                  015D   815 ; This concludes the code that executes in the synchronizing node.
                                  015D   816 ; Here begins the code that executes in the non-synchronizing nodes.
                                  015D   817 ;
                20   BA  015D   818 90$:    POPR    #^M<R5>                         ; Restore CLUFCB addresses
                7D   10  015F   819 95$:    BSBB    DELAY                           ; Wait 0-1 seconds
                                  0161   820 ;
                                  0161   821 ; Send message to synchronizing node telling it that we are ready for
                                  0161   822 ; a command to finish a failover step.
                                  0161   823 ;
          53    24 A5   D0  0161   824 100$:    MOVL    CLUFCB$L_SYNC_CSB(R5),R3 ; CSB address of synchronizing node
                55   DD  0165   825          PUSHL   R5                              ; Save context
                 FE96'   30  0167   826          BSBW    CNX$ALLOC_WARMCDRP_CSB          ; Get a fork block
                 FE93'   30  016A   827          BSBW    CNX$RESOURCE_CHECK              ; Check for pool exhaustion
               ED 50    E9  016D   828          BLBC    R0,90$                          ; Branch if no memory available
                10   BA  0170   829          POPR    #^M<R4>                         ; Restore CLUFCB address into R4
          40 A5   54   D0  0172   830          MOVL    R4,CDRP$L_VAL6(R5)              ; Save CLUFCB address
              44 A5 8ED0  0176   831          POPL    CDRP$L_VAL7(R5)                 ; Completion address
          2C A5   0B   9A  017A   832          MOVZBL  #CLMCNX$K_FNC_READY, -          ; Function code for message
                                  017E   833                  CDRP$L_VAL1(R5)
                4E   10  017E   834          BSBB    300$                            ; Set up CDRP for message
                FE7D'   30  0180   835          BSBW    CNX$SEND_MSG_CSB                ; Send message to remote node
                                  0183   836 ;
                                  0183   837 ;       Resume here when the response message arrives or the connection
                                  0183   838 ;       breaks.
                                  0183   839 ;       Registers contain:
                                  0183   840 ;               R0:     Status
                                  0183   841 ;               R2:     Address of message buffer
                                  0183   842 ;               R3:     Address of CSB
                                  0183   843 ;               R4:     Address of PDT
                                  0183   844 ;               R5:     Address of CDRP
                                  0183   845 ;
              44 A5   DD  0183   846          PUSHL   CDRP$L_VAL7(R5)                 ; Completion address
              30 A5   DD  0186   847          PUSHL   CDRP$L_VAL2(R5)                 ; Failover sequence number
              40 A5   DD  0189   848          PUSHL   CDRP$L_VAL6(R5)                 ; CLUFCB address
               FE71'   30  018C   849          BSBW    CNX$PROCESS_RESPONSE            ; Standard response processing
                20   BA  018F   850          POPR    #^M<R5>                         ; CLUFCB address
        1C A5   8E   D1  0191   851          CMPL    (SP)+,CLUFCB$L_ID(R5)           ; Same step?
                23   12  0195   852          BNEQ    110$                            ; Abandon this failover
           01   50   D1  0197   853          CMPL    R0,#1                           ; Test return status
                C3   1F  019A   854          BLSSU   95$                             ; Branch on NAK
                1C   1A  019C   855          BGTRU   110$                            ; Connection is broken
                                  019E   856 ;
                                  019E   857 ; ACK received
                                  019E   858 ;
    27 20 A5    03   E2  019E   859          BBSS    #CLUFCB$V_FKB_BUSY, -           ; Branch if busy and mark busy
                                  01A3   860                  CLUFCB$L_STATUS(R5),140$
       20 A5    10   C8  01A3   861          BISL2   #CLUFCB$M_WAITING, -            ; Set waiting bit, error if
```

```
                                01A7   862                      CLUFCB$L_STATUS(R5)     ;  already set
        53    6E    D0          01A7   863              MOVL    (SP),R3                 ; Save return PC
                                01AA   864              ASSUME  FKB$L_FR4,EQ,<FKB$L_FR3+4>
    10  A5    53    7D          01AA   865              MOVQ    R3,FKB$L_FR3(R5)        ; Save R3 and R4
0C  A5    BE'AF     9E          01AE   866              MOVAB   B^120$,FKB$L_FPC(R5)    ; Restart PC
        65    55    D0          01B3   867              MOVL    R5,(R5)                 ; Link to self
    04  A5    55    D0          01B6   868              MOVL    R5,4(R5)
        5E    04    C0          01BA   869 110$:        ADDL2   #4,SP                   ; Forget return address
                    05          01BD   870              RSB
                                01BE   871
        53    DD                01BE   872 120$:        PUSHL   R3                      ; Restore caller's address
05  20  A5    03    E5          01C0   873              BBCC    #CLUFCB$V_FKB_BUSY, -   ; Clear busy flag
                                01C5   874                      CLUFCB$L_STATUS(R5),140$
    20  A5    10    CA          01C5   875              BICL2   #CLUFCB$M_WAITING, -    ; Clear waiting bit
                                01C9   876                      CLUFCB$L_STATUS(R5)
                    05          01C9   877              RSB
                                01CA   878
                                01CA   879 140$:        BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
                                01CE   880
                                01CE   881 ;
                                01CE   882 ;       Initialize CDRP for building messages
                                01CE   883 ;
                                01CE   884 ;       R5:     Address of CDRP
                                01CE   885 ;       R4:     Address of CLUFCB
                                01CE   886 ;
4C  A5      04'AF     9E          01CE   887 300$:        MOVAB   B^BLD_STEP_MSG, -       ; Message build routine address
                                01D3   888                      CDRP$L_MSGBLD(R5)
30  A5    1C  A4    D0          01D3   889              MOVL    CLUFCB$L_ID(R4), -      ; Failover sequence number
                                01D8   890                      CDRP$L_VAL2(R5)
34  A5    18  A4    D0          01D8   891              MOVL    CLUFCB$L_STEP(R4), -    ; Failover routine index
                                01DD   892                      CDRP$L_VAL3(R5)
                    05          01DD   893              RSB
                                01DE   894
                                01DE   895 ;
                                01DE   896 ;       Wait 0-1 second on fork and wait queue
                                01DE   897 ;       If a new failover is requested, return to caller's caller
                                01DE   898 ;
                                01DE   899 ;       R5:     Address of CLUFCB fork block
                                01DE   900 ;       R0-R4 are destroyed.
                                01DE   901 ;
                                01DE   902 DELAY:
1D  20  A5    03    E2          01DE   903              BBSS    #CLUFCB$V_FKB_BUSY, -   ; Branch if busy and mark busy
                                01E3   904                      CLUFCB$L_STATUS(R5),20$
    0B  A5    08    90          01E3   905              MOVB    #IPL$_SCS, -            ; Store IPL in fork block
                                01E7   906                      FKB$B_FIPL(R5)
                10    BA        01E7   907              POPR    #^M<R4>                 ; Save return address
                08    BA        01E9   908              POPR    #^M<R3>                 ; Save caller's caller's address
                                01EB   909              FORK_WAIT                       ; Wait and hope some responses appear
                53    DD        01F1   910              PUSHL   R3                      ; Restore caller's caller's address
08  20  A5    03    E5          01F3   911              BBCC    #CLUFCB$V_FKB_BUSY, -   ; Clear busy bit
                                01F8   912                      CLUFCB$L_STATUS(R5),20$
02  20  A5    01    E0          01F8   913              BBS     #CLUFCB$V_PENDING, -    ; Branch if another failover is
                                01FD   914                      CLUFCB$L_STATUS(R5),10$ ;   pending and do it
                54    DD        01FD   915              PUSHL   R4                      ; Restore return address
                    05          01FF   916 10$:         RSB
                                0200   917
                                0200   918 20$:         BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
                        0204    920          .SBTTL  BLD_STEP_MSG - Build Message to Step Failover
                        0204    921
                        0204    922 ;++
                        0204    923 ;
                        0204    924 ; FUNCTIONAL DESCRIPTION:
                        0204    925 ;
                        0204    926 ;       This routine builds the messages used to sequence failover steps
                        0204    927 ;       from data in the CDRP.
                        0204    928 ;
                        0204    929 ; CALLING SEQUENCE:
                        0204    930 ;
                        0204    931 ;       JSB     BLD_STEP_MSG
                        0204    932 ;       IPL is IPL$_SCS
                        0204    933 ;
                        0204    934 ; INPUT PARAMETERS:
                        0204    935 ;
                        0204    936 ;       R2:     Address of message buffer
                        0204    937 ;       R3:     Address of CSB
                        0204    938 ;       R4:     Address of PDT
                        0204    939 ;       R5:     Address of CDRP
                        0204    940 ;               CDRP$L_VAL1(R5): Byte 0 contains facility specific function code
                        0204    941 ;               CDRP$L_VAL2(R5): Failover sequence number
                        0204    942 ;               CDRP$L_VAL3(R5): Failover step index
                        0204    943 ;
                        0204    944 ; OUTPUT PARAMETERS:
                        0204    945 ;
                        0204    946 ;       NONE
                        0204    947 ;
                        0204    948 ; COMPLETION CODES:
                        0204    949 ;
                        0204    950 ;       NONE
                        0204    951 ;
                        0204    952 ; SIDE EFFECTS:
                        0204    953 ;
                        0204    954 ;       R0 and R1 are destroyed
                        0204    955 ;
                        0204    956 ;--
                        0204    957
                        0204    958 BLD_STEP_MSG:
    08 A2    01    90   0204    959          MOVB    #CLSMSG$K_FAC_CNX, -    ; Facility code
                        0208    960                  CLSMSG$B_FACILITY(R2)
09 A2    2C A5    90   0208    961          MOVB    CDRP$L_VAL1(R5), -      ; Facility specific function code
                        020D    962                  CLSMSG$B_FUNC(R2)
0C A2    30 A5    D0   020D    963          MOVL    CDRP$L_VAL2(R5), -      ; Failover sequence number
                        0212    964                  CLMSTP$L_ID(R2)
10 A2    34 A5    D0   0212    965          MOVL    CDRP$L_VAL3(R5), -      ; Failover step index
                        0217    966                  CLMSTP$L_STEP(R2)
                  05    0217    967          RSB
```

```
                     0218   969          .SBTTL  CNX$RCVD_READY - Ready for Failover Step Message Received
                     0218   970
                     0218   971 ;++
                     0218   972 ;
                     0218   973 ; FUNCTIONAL DESCRIPTION:
                     0218   974 ;
                     0218   975 ;       This routine is called when a message from a node ready to perform
                     0218   976 ;       a failover step is received.
                     0218   977 ;
                     0218   978 ; CALLING SEQUENCE:
                     0218   979 ;
                     0218   980 ;       JSB     CNX$RCVD_READY
                     0218   981 ;       IPL is IPL$_SCS
                     0218   982 ;
                     0218   983 ; INPUT PARAMETERS:
                     0218   984 ;
                     0218   985 ;       R2:     Message address
                     0218   986 ;       R3:     CSB of sending system
                     0218   987 ;       R4:     PDT address
                     0218   988 ;       R5:     CDRP address (uninitialized)
                     0218   989 ;
                     0218   990 ; OUTPUT PARAMETERS:
                     0218   991 ;
                     0218   992 ;       NONE
                     0218   993 ;
                     0218   994 ; COMPLETION CODES:
                     0218   995 ;
                     0218   996 ;       NONE
                     0218   997 ;
                     0218   998 ; SIDE EFFECTS:
                     0218   999 ;
                     0218  1000 ;       R0-R5 may be destroyed.
                     0218  1001 ;--
                     0218  1002
                     0218  1003 CNX$RCVD_READY::
          52   DD    0218  1004          PUSHL   R2                              ; Message buffer address
      FDE3'    30    021A  1005          BSBW    CNX$INIT_CDRP                   ; Initialize the CDRP for the response
          04   BA    021D  1006          POPR    #^M<R2>                         ; Restore message buffer address
   54   64 A3   D0    021F  1007          MOVL    CSB$L_CLUB(R3),R4               ; Address of CLUB
54   010C C4   9E    0223  1008          MOVAB   CLUB$B_CLUFCB(R4),R4            ; Address of failover block
          50   D4    0228  1009          CLRL    R0                              ; Assume failure
1C A4    0C A2   D1    022A  1010          CMPL    CLMSTP$L_ID(R2), -              ; Is the expected failover in progress?
                     022F  1011                  CLUFCB$L_ID(R4)
          18   12    022F  1012          BNEQ    20$                             ; Branch if not and ignore message
18 A4    10 A2   D1    0231  1013          CMPL    CLMSTP$L_STEP(R2), -            ; Has this step been passed?
                     0236  1014                  CLUFCB$L_STEP(R4)
          39   1F    0236  1015          BLSSU   90$                             ; Branch if yes -- something is wrong
34 20 A4    02   E1    0238  1016          BBC     #CLUFCB$V_SYNC_NODE, -          ; Branch if this is not the synchronizing
                     023D  1017                  CLUFCB$L_STATUS(R4),90$ ;   node
      50   01   D0    023D  1018          MOVL    #1,R0                           ; Set success
   51   4C A3   3C    0240  1019          MOVZWL  CSB$W_CSID_IDX(R3),R1           ; CSID index of sending system
28 48 A4    51   E2    0244  1020          BBSS    R1, -                           ; Set the response bit corresponding
                     0249  1021                  CLUFCB$B_RESPMAP(R4),90$ ;  to the sending node
          11   BB    0249  1022 20$:     PUSHR   #^M<R0,R4>                      ; Save status and CLUFCB address
   54   64 A3   D0    024B  1023          MOVL    CSB$L_CLUB(R3),R4               ; Address of CLUB
      50   0B   9A    024F  1024          MOVZBL  #CLMCNX$K_FNC_READY,R0          ; Message ID
          FDAB'    30    0252  1025          BSBW    CNX$INIT_STD_RESP               ; Set up response message
```

```
                36 A5   6E   90  0255  1026          MOVB    (SP),CDRP$L_VAL3+2(R5)  ; Store status flag
                     FDA4'   30  0259  1027          BSBW    CNX$RESP_FORGET         ; Send response and exit
                         21  BA  025C  1028          POPR    #^M<R0,R5>              ; Restore status and CLUFCB address
                      0F 50  E9  025E  1029          BLBC    R0,30$                  ; Branch if message NAKed
          0A 20 A5     04   E1  0261  1030          BBC     #CLUFCB$V_WAITING, -    ; Branch if not waiting for responses
                                0266  1031                  CLUFCB$L_STATUS(R5),30$
                                0266  1032          ;
                                0266  1033          ; Synchronizing node waiting for nodes to respond.
                                0266  1034          ; Wake it up.
                                0266  1035          ;
                   55   65   0F  0266  1036          REMQUE  (R5),R5                 ; Dequeue fork block
                                0269  1037          ASSUME  FKB$L_FR4,EQ,<FKB$L_FR3+4>
                53   10 A5   7D  0269  1038          MOVQ    FKB$L_FR3(R5),R3        ; Restore R3 and R4
                     0C B5   16  026D  1039          JSB     @FKB$L_FPC(R5)          ; Call fork routine
                             05  0270  1040 30$:     RSB                             ; Return
                                 0271  1041
                                 0271  1042 90$:     BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
                                0275  1044          .SBTTL  CNX$RCVD_DOSTEP - Do Failover Step Message Received
                                0275  1045
                                0275  1046 ;++
                                0275  1047 ;
                                0275  1048 ; FUNCTIONAL DESCRIPTION:
                                0275  1049 ;
                                0275  1050 ;       This routine is called when a message from the synchronizing node is
                                0275  1051 ;       received requesting that a failover step be performed.
                                0275  1052 ;
                                0275  1053 ; CALLING SEQUENCE:
                                0275  1054 ;
                                0275  1055 ;       JSB     CNX$RCVD_DOSTEP
                                0275  1056 ;       IPL is IPL$_SCS
                                0275  1057 ;
                                0275  1058 ; INPUT PARAMETERS:
                                0275  1059 ;
                                0275  1060 ;       R2:     Message address
                                0275  1061 ;       R3:     CSB of sending system
                                0275  1062 ;       R4:     PDT address
                                0275  1063 ;
                                0275  1064 ; OUTPUT PARAMETERS:
                                0275  1065 ;
                                0275  1066 ;       NONE
                                0275  1067 ;
                                0275  1068 ; COMPLETION CODES:
                                0275  1069 ;
                                0275  1070 ;       NONE
                                0275  1071 ;
                                0275  1072 ; SIDE EFFECTS:
                                0275  1073 ;
                                0275  1074 ;       R0-R5 may be destroyed.
                                0275  1075 ;--
                                0275  1076
                                0275  1077 CNX$RCVD_DOSTEP::
          0C A2    DD           0275  1078          PUSHL   CLMSTP$L_ID(R2)          ; Failover sequence number
          10 A2    DD           0278  1079          PUSHL   CLMSTP$L_STEP(R2)        ; Failover step index
           FD82'   30           027B  1080          BSBW    CNX$DEAL[_MSG_BUF_CSB    ; Deallocate message buffer
    54     64 A3    D0           027E  1081          MOVL    CSB$L_CLUB(R3),R4        ; Address of CLUB
 55    010C C4    9E           0282  1082          MOVAB   CLUB$B_CLUFCB(R4),R5     ; Address of failover block
          03    BA           0287  1083          POPR    #^M<R0,R1>               ; SEQ to R1, INDEX to R0
    1C A5    51    D1           0289  1084          CMPL    R1,CLUFCB$L_ID(R5)       ; Is the expected failover in progress?
          17    12           028D  1085          BNEQ    20$                      ; Branch if not and ignore message
    18 A5    50    D1           028F  1086          CMPL    R0,CLUFCB$L_STEP(R5)     ; Check step number consistency
          12    12           0293  1087          BNEQ    90$                      ; Branch if different -- something is wrong
 0D 20 A5    02    E0           0295  1088          BBS     #CLUFCB$V_SYNC_NODE, -   ; Branch if this is the synchronizing
                                029A  1089                  CLUFCB$L_STATUS(R5),90$ ;   node -- can't happen
 08 20 A5    04    E1           029A  1090          BBC     #CLUFCB$V_WAITING, -     ; Branch if this node not waiting for respon
                                029F  1091                  CLUFCB$L_STATUS(R5),90$
                                029F  1092          ASSUME  FKB$L_FR4,EQ,<FKB$L_FR3+4>
    53    10 A5    7D           029F  1093          MOVQ    FKB$L_FR3(R5),R3         ; Resume fork block
          0C B5    16           02A3  1094          JSB     @FKB$[_FPC(R5)           ;   and continue table
                    05           02A6  1095 20$:     RSB                              ; Return
                                02A7  1096
                                02A7  1097 90$:     BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
```

```
                        02AB  1099          .SBTTL  CNX$FAILO_JNL - Initiate journal recovery
                        02AB  1100
                        02AB  1101 ;++
                        02AB  1102 ;
                        02AB  1103 ; FUNCTIONAL DESCRIPTION:
                        02AB  1104 ;
                        02AB  1105 ;       This routine find and invokes the journal recovery routine.
                        02AB  1106 ;
                        02AB  1107 ; CALLING SEQUENCE:
                        02AB  1108 ;
                        02AB  1109 ;       JSB     CNX$FAILO_JNL
                        02AB  1110 ;       IPL is IPL$_SCS
                        02AB  1111 ;
                        02AB  1112 ; INPUT PARAMETERS:
                        02AB  1113 ;
                        02AB  1114 ;       R5:     Address of Failover Control Block
                        02AB  1115 ;       R4:     Address of CLUB
                        02AB  1116 ;       R3:     Failover ID
                        02AB  1117 ;
                        02AB  1118 ; OUTPUT PARAMETERS:
                        02AB  1119 ;
                        02AB  1120 ;       NONE
                        02AB  1121 ;
                        02AB  1122 ; COMPLETION CODES:
                        02AB  1123 ;
                        02AB  1124 ;       NONE
                        02AB  1125 ;
                        02AB  1126 ; SIDE EFFECTS:
                        02AB  1127 ;
                        02AB  1128 ;       NONE
                        02AB  1129 ;
                        02AB  1130 ;--
                        02AB  1131
                        02AB  1132 CNX$FAILO_JNL::
50   18 A4   D0         02AB  1133          MOVL    CLUB$L_JNL_FAIL(R4),R0  ; Fetch and test routine address
      02   13           02AF  1134          BEQL    10$                     ; No routine found, complete failover step
      60   17           02B1  1135          JMP     (R0)                    ; Jump to routine -- it sees standard
                        02B3  1136                                          ;   failover routine environment
                        02B3  1137
           05           02B3  1138 10$:     RSB
```

```
                        02B4  1140          .SBTTL  CNX$CHECK_FAILOVER - Test for and begin pending failover
                        02B4  1141
                        02B4  1142 ;++
                        02B4  1143 ;
                        02B4  1144 ; FUNCTIONAL DESCRIPTION:
                        02B4  1145 ;
                        02B4  1146 ;       This routine tests for a pending failover and initiates the pending
                        02B4  1147 ;       failover.  If no failover is pending, a check is made for a certain
                        02B4  1148 ;       future failover.  It is is known that a failover will soon happen, the
                        02B4  1149 ;       thread is dropped (return to caller's caller).  If no failover is
                        02B4  1150 ;       pending and no future failover is certain, return to the caller.
                        02B4  1151 ;       It is assumed that failover table processing is in NOREFAIL mode.
                        02B4  1152 ;
                        02B4  1153 ;       This routine is used by failover routines that operate in NOREFAIL
                        02B4  1154 ;       mode to determine whether or not to continue failover processing or
                        02B4  1155 ;       to abandon it in favor of starting over (immediately or in the near
                        02B4  1156 ;       future).  If a decision is made to abandon processing, no return
                        02B4  1157 ;       is made to the caller.
                        02B4  1158 ;
                        02B4  1159 ; CALLING SEQUENCE:
                        02B4  1160 ;
                        02B4  1161 ;       JSB     CNX$CHECK_FAILOVER
                        02B4  1162 ;       IPL is IPL$_SCS
                        02B4  1163 ;       4(SP) : Address of caller's caller
                        02B4  1164 ;
                        02B4  1165 ; INPUT PARAMETERS:
                        02B4  1166 ;
                        02B4  1167 ;       NONE
                        02B4  1168 ;
                        02B4  1169 ; OUTPUT PARAMETERS:
                        02B4  1170 ;
                        02B4  1171 ;       NONE
                        02B4  1172 ;
                        02B4  1173 ; COMPLETION CODES:
                        02B4  1174 ;
                        02B4  1175 ;       NONE
                        02B4  1176 ;
                        02B4  1177 ; SIDE EFFECTS:
                        02B4  1178 ;
                        02B4  1179 ;       If return is to caller, R0 and R1 are destroyed.
                        02B4  1180 ;       If return is to caller's caller, R0-R5 are destroyed.
                        02B4  1181 ;--
                        02B4  1182
                        02B4  1183 CNX$CHECK_FAILOVER::
50   00000000'GF    D0  02B4  1184          MOVL    G^CLU$GL_CLUB,R0        ; Address of CLUB
 1F 012C C0    00    E1  02BB  1185          BBC     #CLUFCB$V_ACTIVE, -     ; Branch if in REFAIL mode and
                        02C1  1186                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0), -
                        02C1  1187                  30$                     ;   bugcheck
 0B 012C C0    01    E1  02C1  1188          BBC     #CLUFCB$V_PENDING, -    ; Branch if no failover is pending
                        02C7  1189                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0), -
                        02C7  1190                  10$
    012C C0    01    CA  02C7  1191          BICL2   #CLUFCB$M_ACTIVE, -     ; Clear failover active bit
                        02CC  1192                  CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0)
          5E    04    C0  02CC  1193          ADDL2   #4,SP                   ; Remove caller's return address
             FD2E    31  02CF  1194          BRW     CNX$MEMBERSHIP_CHANGE   ; Initiate new membership change
                        02D2  1195
  08 1C A0    17    E1  02D2  1196 10$:      BBC     #CLUB$V_LOST_CNX, -     ; Branch if no cluster connections broken
```

```
                     02D7  1197                    CLUB$L_FLAGS(R0),20$
012C C0   01   CA    02D7  1198          BICL2     #CLUFCB$M_ACTIVE, -     ; Clear failover active bit
                     02DC  1199                    CLUB$B_CLUFCB+CLUFCB$L_STATUS(R0)
     5E   04   C0    02DC  1200          ADDL2     #4,SP                   ; Return to caller's caller
                05   02DF  1201 20$:     RSB                               ; No pending failover, return
                     02E0  1202
                     02E0  1203 30$:     BUG_CHECK       CNXMGRERR,FATAL ; Consistency check
                     02E4  1204
                     02E4  1205          .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
_255DUA28:[SYSLOA.OBJ]CLUSTER.MLB;1	1
_255DUA28:[SYS.OBJ]LIB.MLB;1	7
_255DUA28:[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
```

ERRSUBUV1
LIS

ERRSUB790
LIS

ERRSUB780
LIS

ERRSUB750
LIS

ERRSUB730
LIS

FALMAN
LIS

INIADP750
LIS

INIADP730
LIS

INIADP780
LIS

INIADPUVI
LIS

INIADP790
LIS