```
L           JJJ      SSSS
L             J     S    S
L             J     S
L             J      SSS
L             J         S
L   L     J   J     S    S
LLLLL      JJJ      SSSS
```

Sat 22-Mar-1986 20:21:33

Print request number 704

Station: $36

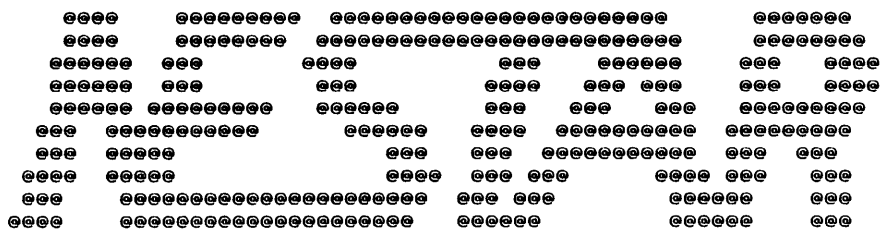
Name: L J Shustek

File Server: BUTLER ($FE)

NFS Pathname:

Filename (s):

Print Server: LENNON ($8A)
Printer: LASER
Setup: LANDSCAPE
Priority: Standard
Copies: 1
Eject: 0



```
ERR  LINE  ADDR
        1                       ;p tab 1,9,17,25,33,41,49,57,65,73
        2                                    ttl    Interrupt-driven Level 2          NESTAR CONFIDENTIAL
        3                       ;-------------------------------------------------------------------------
        4                       ;
        5                       ;file: l2.asm
        6                       ;
        7                       ;
        8                       ;   This is an 68xxx version of interrupt-driven XNS Level 2 routines,
        9                       ;   written for the Level 4 code in C.
       10                       ;
       11                       ;
       12                       ;       The following assumptions are made:
       13                       ;
       14                       ;       1.  The C compiler is Microtec version 5.0.
       15                       ;
       16                       ;       2.  The PIC is located at address 0F0000.
       17                       ;
       18                       ;       3.  The PIC interrupts on level 3.
       19                       ;
       20                       ;       5.  Registers D0, D1, A0, and A1 may be destroyed when called
       21                       ;           from C routines.
       22                       ;
       23                       ;-------------------------------------------------------------------------
       24
       25                       ;       (C) Copyright 1985, 1986, Nestar Systems, Inc.
       26
       27                       ;-------------------------------------------------------------------------------
       28                       ;
       29                       ;       Change log
       30                       ;
       31                       ;  12/ 2/85  L. Shustek   Written in 8088 code for the prototype fileserver L4.
       32                       ;  12/ 6/85  L. Shustek   Add l2_terminate().
       33                       ;  12/15/85  L. Shustek   Fix iret from timer interrupt routine.
       34                       ;  12/19/85  L. Shustek   Reenable rcv interrupts from l2_rcvrelease().
       35                       ;   1/24/86  L. Shustek   Disable interrupts in l2_getbuf().
       36                       ;                         Fix group addressability for C data segment.
       37                       ;   1/28/86  L. Shustek   Enable recon interrupts, and count them.
       38                       ;   1/29/86  L. Shustek   Save es in c_from_int; clobbered if multiple interrupt
       39                       ;   2/ 2/86  L. Shustek   Switch to a fresh stack during interrupt processing.
       40                       ;   2/ 4/86  L. Shustek   Initialize data so we can restart from the top.
       41                       ;   2/ 6/86  L. Shustek   Fix so that no more than one empty transmit buffer
       42                       ;                         at a time is given to L4.
       43                       ;   2/11/86  L. Shustek   Add transmit retries for TA without TMA;  this raises
       44                       ;                         the chance for successful error recovery.  See the
       45                       ;                         commentary in l4private.h about error strategy.
       46                       ;   2/13/86  L. Shustek   Fix interrupt logic to guarantee an edge on the
       47                       ;                         interrupt line.
       48                       ;   2/18/86  L. Shustek   Move general routines into execasm.asm.
       49                       ;   2/21/86  L. Shustek   Disallow broadcast reception temporarily, until
       50                       ;                         supported by L4.
       51                       ;   2/22/86  L. Shustek   Add l2_reverse_xns() to speed up packet processing.
```

```
ERR  LINE  ADDR
       52                          ;   2/25/86  L. Shustek   Have 12_reverse_xns reverse allno too.
       53                          ;----------
       54                          ;   2/28/86  L. Shustek   Convert from 8088 version for the PC to 68xxx version
       55                          ;                         for the PLAN series.  This was done fairly mechanically,
       56                          ;                         so the 68000 code may look a bit strange.
       57                          ;
       58                          ;                         At this point the two source files diverge.  Any
       59                          ;                         algorithmic changes should be made in both versions!
       60                          ;----------
       61                          ;   3/22/86  L. Shustek   Don't take over the PIC interrupt vector now that
       62                          ;                         the EXEC does it.  So change pic_interrupt to
       63                          ;                         N_service, and don't save d0/d1/a0/a1.
       64                          ;
       65                          ;-------------------------------------------------------------------------------
       66
       67
       68                                  opt     -m              ;no macro expansions
       69
       70                          12          idnt
       71
       72                          ;-------------------------------------------------------------------------------
       73                          ;
       74                          ;               General Symbols
       75                          ;
       76                          ;-------------------------------------------------------------------------------
       77
       78
       79 0000000F                TO_XMIT_PKT     equ     15      ; 14-15 * 220 msec = 3 sec packet transmit timeout
       80                                                          ; (Also change same symbol in 14private.h!)
       81
       82 00000001                XMIT_RETRIES    equ     1       ; how many transmit retries if TA without TMA
       83                                                          ; (Also change same symbol in 14private.h!)
       84
       85
       86 000F0000                pic_location    equ     $0f0000 ;location for the PIC
       87
       88 00000003                pic_int_level   equ     3       ;interrupt level for the PIC
       89
       90 00002700                disable         equ     $2700   ;interrupt mask in status register
       91
       92 00000060                int_vectors     equ     $060    ;start of interrupt vectors
       93
```

```
ERR  LINE  ADDR
       95
       96                                ;---------------------------------------------------------------------------
       97                                ;
       98                                ;                Peripheral Interface Card symbols
       99                                ;
      100                                ;---------------------------------------------------------------------------
      101
      102                                ;
      103                                ; RIM command register ("rim_cmd")
      104                                ;
      105
      106 00000000                       broadcast_ok    equ     0       ;do we allow broadcast reception?  0=no, 1=yes
      107
      108 0000001E                       rimcmd_clrflgs  equ     $1e     ;RIM command: clear POR and RECON flags
      109 0000000D                       rimcmd_config_l equ     $0d     ;RIM command: configure long packet mode
      110 00000005                       rimcmd_config_s equ     $05     ;RIM command: configure short packet mode
      111 00000004                       rimcmd_recv_en  equ     $04 + $80 * broadcast_ok
      112                                                                ;RIM command: receive enable, add 8*buffer#
      113 00000003                       rimcmd_xmit_en  equ     $03     ;RIM command: transmit enable, add 8*buffer#
      114 00000001                       rimcmd_xmit_dis equ     $01     ;RIM command: disable transmit
      115
      116
      117                                ;
      118                                ; RIM status register ("rim_stat") and interrupt mask register ("rim_int")
      119                                ;
      120
      121 00000080                       ri              equ     $80     ;RIM status: receiver inhibited
      122 00000010                       por             equ     $10     ;RIM status: power-on reset occurred
      123 00000004                       recon           equ     $04     ;RIM status: recon occurred
      124 00000002                       tma             equ     $02     ;RIM status: transmit message acknowledged
      125 00000001                       ta              equ     $01     ;RIM status: transmitter available
      126
      127
      128                                ;
      129                                ; NIC control register ("nic_ctl")
      130                                ;
      131
      132 00000001                       rim_int_enb     equ     $01     ;enable RIM interrupts
      133
      134
      135                                ;
      136                                ; NIC status register ("nic_stat")
      137                                ;
      138
      139 00000080                       rim_interrupt   equ     $80     ;RIM is interrupting
      140
```

```
ERR  LINE  ADDR
      142
      143                              ;
      144                              ;       Packet buffer states
      145                              ;
      146                              ; Of the four Arcnet RIM buffers, the first two are used as
      147                              ; transmit buffers and the second two as receive buffers.
      148                              ; Each proceeds through the appropriate four states in sequence.
      149                              ; Buffer data structures are two bytes wide and indexed by 0,2,4,6.
      150                              ; We use the trick of XORing the buffer index to reference the "other"
      151                              ; transmit or receive buffer.
      152                              ;
      153
      154 00000000                     buf0            equ     0       ;xmit buffer     ;buffer indices
      155 00000002                     buf1            equ     2       ;xmit buffer
      156 00000004                     buf2            equ     4       ;rcv buffer
      157 00000006                     buf3            equ     6       ;rcv buffer
      158
      159 00000002                     other_buffer    equ     2       ;"other rcv/xmit buffer" XOR symbol
      160
      161 00000001                     xmit_empty      equ     1       ;transmit buffer empty
      162 00000002                     xmit_in14       equ     2       ;transmit buffer given to Level 4 to fill
      163 00000003                     xmit_full       equ     3       ;transmit buffer full, awaiting transmission
      164 00000004                     xmit_enabled    equ     4       ;transmit buffer full, enabled for transmission
      165
      166 00000005                     rcv_empty       equ     5       ;receive buffer empty
      167 00000006                     rcv_enabled     equ     6       ;receive buffer empty, enabled for reception
      168 00000007                     rcv_full        equ     7       ;receive buffer full, awaiting processing
      169 00000008                     rcv_in14        equ     8       ;receive buffer given to Level 4 to empty
      170
```

```
ERR  LINE  ADDR
      172
      173                               ;
      174                               ;               Map of the PIC
      175                               ;
      176                               ;
      177                               ;       This is an absolute section, but generates no code or data.
      178                               ;       The Microtec assembler doesn't seem to have a dummy section type.
      179                               ;
      180
      181                                       org     pic_location
      182
      183 000F0000                      pic_map         ds.b    0
      184
      185 000F0000                      prom            ds.b    32              ;ID prom
      186 000F0007                      prom_nw_addr    equ     prom+7          ;location of XNS address in prom
      187
      188                                       org     pic_location+$800
      189 000F0800                      int_cond        ds.b    1               ;interrupt condition register
      190 000F0801                      nic_ctl         ds.b    0               ;network interface control register
      191 000F0801                      nic_stat        ds.b    1               ;network interface status register
      192
      193                                       org     pic_location+$3000
      194 000F3000                      page0           ds.b    512             ;buffer 0
      195 000F3200                      page1           ds.b    512             ;buffer 1
      196 000F3400                      page2           ds.b    512             ;buffer 2
      197 000F3600                      page3           ds.b    512             ;buffer 3
      198
      199 000F3800                      rim_int         ds.b    0               ; RIM interrupt mask register (write)
      200 000F3800                      rim_stat        ds.b    1               ; RIM status register (read)
      201 000F3801                      rim_cmd         ds.b    1               ; RIM command register (write)
      202 000F3802                      arc_address     ds.b    1               ;Arcnet station address (read)
      203
```

```
ERR  LINE  ADDR
      205
      206                               ;-------------------------------------------------------------------------
      207                               ;
      208                               ;               DATA
      209                               ;
      210                               ;-------------------------------------------------------------------------
      211
      212                                               section 14              ;in the C data section
      213
      214
      215                               ;
      216                               ;               Global static data segment
      217                               ;               --------------------------
      218                               ;
      219
      220                                               xref    .l4cnt_l2int_ri         ;count of RI interrupts
      221                                               xref    .l4cnt_l2int_ta         ;count of TA interrupts
      222                                               xref    .l4cnt_l2int_recon      ;count of recon interrupts
      223                                               xref    .l4cnt_xmitnoack;count of TA w/o TMA
      224                                               xref    .l4cnt_xmittimeout      ;count of transmit timeouts
      225                                               xref    .l4cnt_p_retries;count of packet xmit retries
      226
      227
      228                               ;
      229                               ;               External function references
      230                               ;               ----------------------------
      231                               ;
      232
      233                                               xref    .l4_timerint    ;Level 4 timer interrupt routine
      234                                               xref    .l4_rcvintr     ;Level 4 receive interrupt routine
      235                                               xref    .l4_gotbuf      ;Level 4 transmit buffer avail int rtn
      236                                               xref    .panic          ;failed assertion routine
      237
      238
      239                               ;
      240                               ;               Messages
      241                               ;               --------
      242                               ;
      243
      244 00000000 6C32 5F73 656E       panic1          dc.b    'l2_sendbuf: bad buf addr',0
      244 00000006 6462 7566 3A20
      244 0000000C 6261 6420 6275
      244 00000012 6620 6164 6472
      244 00000018 00
      245 00000019 6C32 5F73 656E       panic2          dc.b    'l2_sendbuf: bad buf state',0
      245 0000001F 6462 7566 3A20
      245 00000025 6261 6420 6275
      245 0000002B 6620 7374 6174
      245 00000031 6500
      246 00000033 6C32 5F72 6376       panic3          dc.b    'l2_rcvrelease: bad buf addr',0
```

```
ERR  LINE  ADDR
      246 00000039 7265 6C65 6173
      246 0000003F 653A 2062 6164
      246 00000045 2062 7566 2061
      246 0000004B 6464 7200
      247 0000004F 6C32 5F72 6376       panic4          dc.b    'l2_rcvrelease: bad buf state 1',0
      247 00000055 7265 6C65 6173
      247 0000005B 653A 2062 6164
      247 00000061 2062 7566 2073
      247 00000067 7461 7465 2031
      247 0000006D 00
      248 0000006E 6C32 5F72 6376       panic5          dc.b    'l2_rcvrelease: bad buf state 2',0
      248 00000074 7265 6C65 6173
      248 0000007A 653A 2062 6164
      248 00000080 2062 7566 2073
      248 00000086 7461 7465 2032
      248 0000008C 00
      249
      250
      251                               ;
      252                               ;               Local data
      253                               ;               ----------
      254                               ;
      255
      256 0000008E                      local_data      ds      0
      257
      258 0000008E 4E45 5354 4152       nestar          dc.b    'NESTAR'
      259
      260 00000094                                      ds      0
      261 00000094 0000 0000 0000       xnsaddr         dc.w    0,0,0           ;our 6-byte XNS address
      262
      263 0000009A                      picintsv        ds.l    1               ;previous pic interrupt vector
      264 0000009E                      timintsv        ds.l    1               ;previous timer interrupt vector
      265
      266 000000A2 00                   xmit_timer      dc.b    0               ;transmit packet timer
      267 000000A3 00                   n_xmit_retries  dc.b    0               ;transmit retry count
      268 000000A4 00                   n_owed_buffs    dc.b    0               ;number of buffers we owe to l4
      269 000000A5 00                   rim_int_copy    dc.b    0               ;copy of rim_int, the interrupt mask
      270
```

```
ERR  LINE  ADDR
      272
      273                              ;               Buffer control
      274
      275
      276 000000A6                     .l2_buff_state  ds      0               ;external defn for debugging print
      277                                              xdef    .l2_buff_state
      278
      279 000000A6 0100                buff_state      dc.b    xmit_empty,0    ;transmit buffer #0 starts empty
      280 000000A8 0100                                dc.b    xmit_empty,0    ;transmit buffer #1 starts empty
      281 000000AA 0500                                dc.b    rcv_empty,0     ;receive buffer #2 starts empty
      282 000000AC 0500                                dc.b    rcv_empty,0     ;receive buffer #3 starts empty
      283
      284 000000AE 0300                enable_cmds     dc.b    rimcmd_xmit_en+8*0,0   ;buffer xmit or rcv enable cmds
      285 000000B0 0B00                                dc.b    rimcmd_xmit_en+8*1,0
      286 000000B2 1400                                dc.b    rimcmd_recv_en+8*2,0
      287 000000B4 1C00                                dc.b    rimcmd_recv_en+8*3,0
      288
      289 000000B6 3000                rim_buf_offset  dc.w    page0-pic_map           ;offsets to rim buffers
      290 000000B8 3200                                dc.w    page1-pic_map
      291 000000BA 3400                                dc.w    page2-pic_map
      292 000000BC 3600                                dc.w    page3-pic_map
      293
      294
      295
      296                              ;
      297                              ;               Macro to click the speaker
      298                              ;
      299                              ; Destroys al.
      300                              ;
      301
      302                              click           macro
      303                              ;                               ;We ain't got none, unfortunately!
      304                                              endm
      305
      306
```

```
ERR  LINE  ADDR
      308
      309
      310                              ;-------------------------------------------------------------------------
      311                              ;
      312                              ;       boolean l2_init ( &our_addr )
      313                              ;
      314                              ;               short int our_addr[3]
      315                              ;
      316                              ;
      317                              ;       Initialize Level 2 and return the 6-byte host station address.
      318                              ;       Return TRUE if initialization succeeded.
      319                              ;
      320                              ;-------------------------------------------------------------------------
      321
      322                                      xdef    .l2_init
      323                              .l2_init:
      324 000000BE 206F 0004                   move.l  4(sp),a0                ; &our_addr in a0 forever....
      325 000000C2 48E7 003C                   movem.l a2/a3/a4/a5,-(sp)       ; push a2-a5
      326          45F9 000F 0000              using!  pic_map,a2              ; a2 points to pic
      327          47FA FFC0                   using!  local_data,a3           ; a3 points to local data
      328
      329
      330                              ;
      331                              ;               Miscellaneous data re-initialization
      332                              ;
      333
      334 000000D0 177C 0000 0016              move.b  #0,n_owed_buffs                 ;we owe no buffers
      335
      336 000000D6 177C 0001 0018              move.b  #xmit_empty,buff_state+buf0     ;transmit buffer #0 starts empty
      337 000000DC 177C 0001 001A              move.b  #xmit_empty,buff_state+buf1     ;transmit buffer #1 starts empty
      338 000000E2 177C 0005 001C              move.b  #rcv_empty,buff_state+buf2      ;receive buffer #2 starts empty
      339 000000E8 177C 0005 001E              move.b  #rcv_empty,buff_state+buf3      ;receive buffer #3 starts empty
      340
      341                              ;
      342                              ;               Check that the PIC exists
      343                              ;
      344
      345 000000EE 303C 0005                   move    #5,d0
      346 000000F2 49EA 0000                   lea     prom,a4
      347 000000F6 4BEB 0000                   lea     nestar,a5               ;compare first 6 bytes
      348 000000FA BB0C                pic_check:      cmp.b   (a4)+,(a5)+             ;to "NESTAR"
      349 000000FC 57C8 FFFC                   dbeq    d0,pic_check
      350 00000100 6658 4E71                   bne     error
      351
      352
      353                              ;
      354                              ;               Get our host address
      355                              ;
      356
      357 00000104 323C 0000                   move    #0,d1
```

```
ERR  LINE  ADDR
      358 00000108 49EA 0007                        lea     prom_nw_addr,a4         ;serial number on nic
      359 0000010C 4BEB 0006                        lea     xnsaddr,a5              ;where to store it locally
      360                                   repeat
      361 00000110 101C                             move.b  (a4)+,d0                ;get from PIC prom
      362 00000112 1AC0                             move.b  d0,(a5)+                ;save locally
      363 00000114 1180 1000                        move.b  d0,0(a0,d1)             ;save in our_addr
      364 00000118 5241                             add     #1,d1
      365                                   until  d1 <eq> #5
      366
      367 00000120 102A 3802                        move.b  arc_address,d0          ;last byte is arcnet address
      368 00000124 1AC0                             move.b  d0,(a5)+                ;save locally
      369 00000126 1180 1000                        move.b  d0,0(a0,d1)             ;save in our_addr
      370
      371
      372                               ;
      373                               ;               Initialize the NIC
      374                               ;
      375
      376
      377 0000012A 157C 001E 3801                   move.b  #rimcmd_clrflgs,rim_cmd ;clear POR and RECON flags
      378 00000130 157C 000D 3801                   move.b  #rimcmd_config_1,rim_cmd;configure for long packets
      379
      380
      381                               ;
      382                               ;               Setup the NIC interrupt routine
      383                               ;
      384                               ;       ;;;     Don't need to anymore: EXEC does it.
      385
      386                                       ;;;     move.l  int_vectors+pic_int_level*4,a1  ;save previous vector
      387                                       ;;;     move.l  a1,picintsv
      388                                       ;;;     lea     pic_interrupt,a1
      389                                       ;;;     move.l  a1,int_vectors+pic_int_level*4  ;setup ours
      390
      391
      392                               ;
      393                               ;               Setup the timer interrupt routine
      394                               ;
      395
      396                                       ;nothing yet
      397
      398
      399                               ;
      400                               ;               Enable NIC interrupts and start a receive command
      401                               ;
      402
      403
      404 00000136 157C 0001 0801                   move.b  #rim_int_enb,nic_ctl    ;enable nic interrupts
      405 0000013C 122B 0024                        move.b  enable_cmds+buf2,d1     ;enable rcv on buffer 2
      406 00000140 1541 3801                        move.b  d1,rim_cmd
      407 00000144 177C 0006 001C                   move.b  #rcv_enabled,buff_state+buf2    ;change its state
      408 0000014A 157C 0084 3800                   move.b  #ri+recon,rim_int               ;enable rcv and recon interrupts
```

```
ERR  LINE  ADDR
      409 00000150 177C 0084 0017                    move.b  #ri+recon,rim_int_copy
      410 00000156 600A 4E71                         bra     init_ok
      411
      412                                 ;
      413                                 ;          Return to the caller with the host address.
      414                                 ;
      415
      416 0000015A 303C 0000              error:     move    #0,d0           ;return d0 = FALSE if error
      417 0000015E 6006 4E71                         bra     init_exit
      418
      419 00000162 303C 0001              init_ok:   move    #1,d0          ;return d0 = TRUE if no error
      420
      421
      422                                 init_exit:
      423 00000166 4CDF 3C00                         movem.l (sp)+,a5/a4/a3/a2       ;restore a2-a5
      424 0000016A 4E75                              rts                             ;return with d0 set
      425
      426                                            endu    a2,a3
      427
```

```
ERR  LINE  ADDR
      429
      430                              ;-------------------------------------------------------------------------
      431                              ;
      432                              ;       l2_terminate ()
      433                              ;
      434                              ;  Close down level 2.
      435                              ;  Remove all interrupt routines.
      436                              ;
      437                              ;-------------------------------------------------------------------------
      438
      439                                              xdef    .l2_terminate
      440                              .l2_terminate:
      441 0000016C 48E7 0030                           movem.l a2/a3,-(sp)             ; push a2, a3
      442          45F9 000F 0000                      using!  pic_map,a2              ; a2 points to pic
      443          47FA FF16                           using!  local_data,a3           ; a3 points to local data
      444
      445 0000017A 157C 0000 0801                      move.b  #0,nic_ctl              ;disable network interrupts
      446 00000180 157C 0000 3800                      move.b  #0,rim_int
      447
      448                                      ;;;;    move.l  picintsv,a1             ;restore previous int vector
      449                                      ;;;;    move.l  a1,int_vectors+pic_int_level*4
      450
      451                                              ;       must restore timer interrupt here
      452
      453 00000186 4CDF 0C00                           movem.l (sp)+,a3/a2     ;restore a2, a3
      454 0000018A 4E75                                rts                     ;return
      455
      456   .                                          endu    a2,a3
      457
```

```
ERR  LINE  ADDR
      459
      460                              ;-------------------------------------------------------------------------
      461                              ;
      462                              ;       faraddr _l2_getbuf ()
      463                              ;
      464                              ; Get an empty transmit buffer and return a long pointer to it.
      465                              ;
      466                              ; If there is no free buffer, return NIL and sometime later call
      467                              ; l4_gotbuf( faraddr ) as an interrupt routine and provide it the
      468                              ; long pointer to the buffer.  Only one call to l4_gotbuf() should
      469                              ; be outstanding until the buffer is freed with a call to l2_sendbuf().
      470                              ;
      471                              ;  Arcnet logic:
      472                              ;
      473                              ;       IF buffer_0 is empty, return buffer_0;
      474                              ;       IF buffer_1 is empty, return buffer_1;
      475                              ;       OTHERWISE  ++ n_owed_buffs;
      476                              ;                  return NIL;
      477                              ;
      478                              ;-------------------------------------------------------------------------
      479
      480                                       xdef    .l2_getbuf
      481                              .l2_getbuf:
      482 0000018C 48E7 0030                    movem.l a2/a3,-(sp)             ; push a2, a3
      483          45F9 000F 0000               using!  pic_map,a2              ; a2 points to pic
      484          47FA FEF6                    using!  local_data,a3           ; a3 points to local data
      485
      486 0000019A 40E7                         move.w  sr,-(sp)                ;-------
      487 0000019C 007C 2700                    or      #disable,sr             ;------- disable interrupts
      488
      489 000001A0 0C2B 0001 0018               cmp.b   #xmit_empty,buff_state+buf0     ;is buffer 0 available?
      490                                       if <eq>   then.s
      491 000001A8 177C 0002 0018                 move.b  #xmit_in14,buff_state+buf0    ;yes: use it
      492 000001AE 203C 000F 3000                 move.l  #page0,d0
      493
      494                                       else.s
      495 000001B6 0C2B 0001 001A               cmp.b   #xmit_empty,buff_state+buf1     ;is buffer 1 available?
      496                                       if <eq>   then.s
      497 000001BE 177C 0002 001A                 move.b  #xmit_in14,buff_state+buf1    ;yes: use it
      498 000001C4 203C 000F 3200                 move.l  #page1,d0
      499
      500                                       else.s
      501 000001CC 522B 0016                      add.b  #1,n_owed_buffs        ;no buffer: remember request
      502 000001D0 7000                           move.l #0,d0                          ;and return NIL
      503
      504                                       endi
      505                                       endi
      506
      507 000001D2 46DF                         move.w  (sp)+,sr                ;------ restore interrupts
      508
```

```
ERR  LINE  ADDR
      509 000001D4 4CDF 0C00                    movem.l (sp)+,a2/a3          ;restore a2, a3
      510 000001D8 4E75                         rts                          ;return
      511
      512                                       endu    a2,a3
      513
```

```
ERR  LINE  ADDR
      515
      516                              ;-------------------------------------------------------------------------
      517                              ;
      518                              ;       l2_sendbuf ( faraddr buffer)
      519                              ;
      520                              ; Send, or queue for sending, the transmit buffer whose address is
      521                              ; supplied.  This must be a buffer previously supplied by l2_getbuf()
      522                              ; or to l4_gotbuf();
      523                              ;
      524                              ; If we have an empty buffer and still owe one to L4, give it him now.
      525                              ; This is because L4 can only handle one empty transmit buffer at a time.
      526                              ;
      527                              ; Arcnet logic:
      528                              ;       assert: buffer is buffer_0 or buffer_1
      529                              ;       assert: buffer status is "in L4".
      530                              ;       IF the status of the other buffer is "transmitting"
      531                              ;         THEN  set buffer status to "awaiting transmit"
      532                              ;         ELSE  start transmitting this buffer;
      533                              ;               set status to "transmitting";
      534                              ;               IF n_owed_buffs>0
      535                              ;               AND the status of the other buffer is "empty"
      536                              ;                 THEN --n_owed_buffer;
      537                              ;                      l4_gotbuf(the other buffer)
      538                              ;
      539                              ;-------------------------------------------------------------------------
      540
      541                                       xdef    .l2_sendbuf
      542                              .l2_sendbuf:
      543 000001DA 206F 0004                    move.l  4(sp),a0                ; buffer address in a0 always
      544 000001DE 48E7 0030                    movem.l a2/a3,-(sp)             ; push a2, a3
      545          45F9 000F 0000               using!  pic_map,a2              ; a2 points to pic
      546          47FA FEA4                    using!  local_data,a3           ; a3 points to local data
      547
      548 000001EC B1FC 000F 3000              cmp.l   #page0,a0               ;better be page 0
      549                                       if <eq> then.s
      550 000001F4 303C 0000                     move   #buf0,d0
      551                                       else.s
      552 000001FA B1FC 000F 3200              cmp.l   #page1,a0               ;or page 1
      553                                       if <eq> then.s
      554 00000202 303C 0002                     move   #buf1,d0
      555                                       else.s
      556 00000208 486B FF72                     pea    panic1                  ;neither: panic("bad buf addr")
      557 0000020C 4EB9 0000 0000 E              jsr    .panic
      558                                       endi
      559                                       endi
      560
      561 00000212 0C33 0002 0018              cmp.b   #xmit_in14,buff_state(d0) ;assert buffer was in 14
      562                                       if <ne> then.s
      563 0000021A 486B FF8B                     pea    panic2
      564 0000021E 4EB9 0000 0000 E              jsr    .panic           ;panic("bad buf state")
```

```
ERR  LINE  ADDR
      565                                   endi
      566
      567 00000224 40E7                     move.w  sr,-(sp)                ;-------
      568 00000226 007C 2700                or      #disable,sr             ;------- disable interrupts
      569
      570 0000022A 0A40 0002                eor     #other_buffer,d0        ;look at the other buffer
      571 0000022E 0C33 0004 0018           cmp.b   #xmit_enabled,buff_state(d0)    ;sending?
      572
      573                                   if <eq> then.s                          ;yes: xmitter is busy
      574 00000236 0A40 0002                  eor     #other_buffer,d0              ;switch back to our buffer
      575 0000023A 17BC 0003 0018             move.b  #xmit_full,buff_state(d0)     ;state is "awaiting xmit"
      576
      577                                   else.s                                  ;xmitter is free
      578 00000242 0A40 0002                  eor     #other_buffer,d0              ;switch back to our buffer
      579 00000246 1233 0020                  move.b  enable_cmds(d0),d1
      580 0000024A 1541 3801                  move.b  d1,rim_cmd
      581 0000024E 177C 000F 0014             move.b  #TO_XMIT_PKT,xmit_timer       ;start transmit timer
      582 00000254 177C 0000 0015             move.b  #0,n_xmit_retries             ;start retry counter
      583 0000025A 17BC 0004 0018             move.b  #xmit_enabled,buff_state(d0)  ;"we are sending"
      584 00000260 002B 0001 0017             or.b    #ta,rim_int_copy
      585 00000266 122B 0017                  move.b  rim_int_copy,d1               ;enable transmit interrupts
      586 0000026A 1541 3800                  move.b  d1,rim_int
      587
      588 0000026E 0C2B 0000 0016             cmp.b  #0,n_owed_buffs                  ;if we owe buffers
      589                                     if <hi> then.s
      590 00000276 0A40 0002                    eor    #other_buffer,d0
      591 0000027A 0C33 0001 0018               cmp.b  #xmit_empty,buff_state(d0)     ; and the other buffer
      592                                       if <eq> then.s
      593 00000282 532B 0016                        sub.b  #1,n_owed_buffs            ; then reduce the coun
      594 00000286 17BC 0002 0018                   move.b #xmit_in14,buff_state(d0)  ; and give that buffer
      595 0000028C 3233 0028                        move.w  rim_buf_offset(d0),d1
      596 00000290 4872 1000                        pea     0(a2,d1.w) ;pic_map(d1.w)
      597 00000294 4EB9 0000 0000 E                 jsr     .14_gotbuf              ;14_gotbuf (faraddr)
      598 0000029A 588F                             add.l   #4,sp
      599                                       endi
      600                                     endi
      601                                   endi
      602
      603 0000029C 46DF                     move.w  (sp)+,sr                ;------ restore interrupts
      604
      605 0000029E 4CDF 0C00                movem.l (sp)+,a2/a3             ;restore a2, a3
      606 000002A2 4E75                     rts                             ;return
      607
      608                                   endu    a2,a3
      609
```

```
ERR  LINE  ADDR
      611
      612                              ;---------------------------------------------------------------------
      613                              ;
      614                              ;       l2_rcvrelease ( faraddr buffer )
      615                              ;
      616                              ;  Release the received packet whose buffer address is supplied.
      617                              ;  It was previously provided by a call to l4_rcvintr().
      618                              ;  This can cause l4_rcvintr() to be called if another packet is ready.
      619                              ;
      620                              ;  Arcnet logic:
      621                              ;       assert: buffer is buffer_2 or buffer_3
      622                              ;       assert: buffer status was "in L4"
      623                              ;       IF the other buffer is not receiving
      624                              ;         THEN  enable receive on this buffer
      625                              ;               assert: other buffer is full of data
      626                              ;               l4_rcvintr (other buffer)
      627                              ;
      628                              ;---------------------------------------------------------------------
      629
      630
      631
      632                                              xdef    .l2_rcvrelease
      633                              .l2_rcvrelease:
      634 000002A4 206F 0004                           move.l  4(sp),a0                ; buffer address in a0 always
      635 000002A8 48E7 0030                           movem.l a2/a3,-(sp)             ; push a2, a3
      636          45F9 000F 0000                      using!  pic_map,a2              ; a2 points to pic
      637          47FA FDDA                           using!  local_data,a3           ; a3 points to local data
      638
      639 000002B6 B1FC 000F 3400                      cmp.l   #page2,a0               ;better be page 2
      640                                              if <eq> then.s
      641 000002BE 303C 0004                             move  #buf2,d0
      642                                              else.s
      643 000002C4 B1FC 000F 3600                      cmp.l   #page3,a0               ;or page 3
      644                                              if <eq> then.s
      645 000002CC 303C 0006                             move  #buf3,d0
      646                                              else.s
      647 000002D2 486B FFA5                             pea   panic3                  ;neither: panic("bad buf addr")
      648 000002D6 4EB9 0000 0000 E                      jsr   .panic
      649                                              endi
      650                                              endi
      651
      652 000002DC 0C33 0008 0018                      cmp.b   #rcv_in14,buff_state(d0) ;assert buffer was in 14
      653                                              if <ne> then.s
      654 000002E4 486B FFC1                             pea   panic4
      655 000002E8 4EB9 0000 0000 E                      jsr   .panic          ;panic("bad buf state")
      656                                              endi
      657
      658 000002EE 40E7                                move.w  sr,-(sp)                ;-------
      659 000002F0 007C 2700                           or      #disable,sr             ;------- disable interrupts
      660
```

```
ERR  LINE  ADDR
      661 000002F4 17BC 0005 0018                    move.b  #rcv_empty,buff_state(d0)       ;buffer is now empty
      662
      663 000002FA 0A40 0002                         eor     #other_buffer,d0;look at the other rcv buffer
      664 000002FE 0C33 0006 0018                    cmp.b   #rcv_enabled,buff_state(d0)
      665
      666                                            if <ne> then.s                  ;if it is not enabled for rcv
      667                                                                            ;then it must be full of data
      668 00000306 0A40 0002                           eor     #other_buffer,d0      ;switch back to buffer just freed
      669 0000030A 1233 0020                           move.b  enable_cmds(d0),d1
      670 0000030E 1541 3801                           move.b  d1,rim_cmd                    ;enable receive
      671 00000312 17BC 0006 0018                      move.b  #rcv_enabled,buff_state(d0)   ;"we are receiving"
      672 00000318 002B 0080 0017                      or.b    #ri,rim_int_copy              ;enable rcv interrupts
      673 0000031E 122B 0017                           move.b  rim_int_copy,d1
      674 00000322 1541 3800                           move.b  d1,rim_int
      675
      676 00000326 0A40 0002                           eor     #other_buffer,d0              ;switch back to other rcv buffer
      677 0000032A 0C33 0007 0018                      cmp.b   #rcv_full,buff_state(d0)      ;assert it is full
      678                                              if <ne> then.s
      679 00000332 486B FFE0                             pea   panic5
      680 00000336 4EB9 0000 0000 E                      jsr   .panic
      681                                              endi
      682 0000033C 17BC 0008 0018                      move.b  #rcv_in14,buff_state(d0)      ;give that buffer to 14
      683 00000342 3233 0028                           move.w  rim_buf_offset(d0),d1
      684 00000346 4872 1000                           pea     0(a2,d1.w) ;pic_map(d1.w)
      685 0000034A 4EB9 0000 0000 E                    jsr     .14_rcvintr                   ;14_rcvintr (faraddr)
      686 00000350 588F                                add.l   #4,sp
      687                                            endi
      688   .
      689 00000352 46DF                              move.w  (sp)+,sr                ;------ restore interrupts
      690
      691 00000354 4CDF 0C00                         movem.l (sp)+,a3/a2             ;restore a2, a3
      692 00000358 4E75                              rts                             ;return
      693
      694                                            endu    a2,a3
```

```
ERR  LINE  ADDR
      696
      697                                ;-------------------------------------------------------------------------
      698                                ;
      699                                ;
      700                                ;
      701                                ;                The NIC interrupt routine
      702                                ;
      703                                ;
      704                                ;
      705                                ;  This is a hardware interrupt routine, so NO registers can be changed,
      706                                ;  and NO assumptions can be made about their contents.
      707                                ;
      708                                ;
      709                                ;  If this is a "receive done" interrupt (RI just came on) then
      710                                ;       Mark the buffer as "full".
      711                                ;       If the other buffer is empty, enable receive on it.
      712                                ;       If the other buffer is not in L4, call l4_rcvintr and mark
      713                                ;          this buffer as "in l4".
      714                                ;
      715                                ;  If this is a "transmit done" interrupt (TA just came on) then
      716                                ;       Mark the buffer being transmitted as empty.
      717                                ;       If we got TA without TMA and we are eligible for a retry,
      718                                ;          restart transmission and exit.
      719                                ;       If the other buffer is full, start transmitting it,
      720                                ;          otherwise disable the transmitter interrupt.
      721                                ;       If we owe L4 a transmit buffer and the other buffer
      722                                ;       isn't already given to L4, call l4_gotbuf().
      723                                ;
      724                                ;  If this is a recon (or, impossibly, a power-on reset) interrupt,
      725                                ;  just clear the condition and increment a counter.
      726                                ;
      727                                ;  In high traffic situations it is fairly important to process receive
      728                                ;  interrupts before transmit interrupts.  If you do the reverse, and
      729                                ;  processing the transmit interrupt schedules more transmissions, the
      730                                ;  receive buffers can be backed up to the point where the other stations
      731                                ;  will timeout their transmission to us.
      732                                ;
      733                                ;-------------------------------------------------------------------------
      734
      735                                                xdef    pic_interrupt   ;for debugging only
      736                                                xdef    .N_service      ;what EXEC calls for a RIM interrupt
      737                                pic_interrupt:
      738                                .N_service:
      739                                        ;;;;    movem.l d0/d1/a0/a1/a2/a3,-(sp) ;working registers
      740 0000035A 48E7 0030                         movem.l a2/a3,-(sp)             ; EXEC saved d0/d1/a0/a1
      741
      742 0000035E                                   click                           ;click the speaker
      743
      744          45F9 000F 0000                    using!  pic_map,a2              ; a2 points to pic
      745          47FA FD28                         using!  local_data,a3           ; a3 points to local data
```

ERR LINE ADDR
      746

```
ERR  LINE  ADDR
      748
      749                              check_interrupts:
      750
      751                              ;
      752                              ;              Receive interrupt processing
      753                              ;
      754
      755 00000368 122A 3800                          move.b  rim_stat,d1             ;which interrupt bits
      756 0000036C C22B 0017                          and.b   rim_int_copy,d1         ;mask says which are enabled
      757 00000370 0201 0080                          and.b   #ri,d1
      758                                             if <ne> then.s                  ;------ RI interrupt -----
      759
      760 00000376 52B9 0000 0000 E                     add.l #1,.l4cnt_l2int_ri              ;count RI interrupt
      761
      762 0000037C 303C 0004                            move  #buf2,d0                        ;get index to which buffer
      763 00000380 0C33 0006 0018                       cmp.b #rcv_enabled,buff_state(d0)
      764                                               if <ne> then.s
      765 00000388 303C 0006                              move #buf3,d0
      766                                               endi
      767 0000038C 17BC 0007 0018                       move.b #rcv_full,buff_state(d0)       ;mark it full
      768
      769 00000392 0A40 0002                            eor   #other_buffer,d0        ;look at the other rcv buffer
      770 00000396 0C33 0005 0018                       cmp.b #rcv_empty,buff_state(d0)
      771                                               if <eq> then.s                        ;if it is empty
      772 0000039E 1233 0020                              move.b enable_cmds(d0),d1
      773 000003A2 1541 3801                              move.b d1,rim_cmd                 ;enable receive
      774 000003A6 17BC 0006 0018                         move.b #rcv_enabled,buff_state(d0)  ;"it is receiving"
      775                                                ;(rcv interrupts are already enabled.)
      776                                               else.s
      777 000003AE 022B 007F 0017                         and.b   #255-ri,rim_int_copy
      778 000003B4 122B 0017                              move.b  rim_int_copy,d1           ;disable rcvr interrupts
      779 000003B8 1541 3800                              move.b  d1,rim_int
      780                                               endi
      781
      782 000003BC 0C33 0008 0018                       cmp.b  #rcv_inl4,buff_state(d0)       ;is the other buffer in l4?
      783                                               if <ne> then.s                        ;if not,
      784 000003C4 0A40 0002                              eor     #other_buffer,d0            ;switch to the newly rcvd buffer
      785 000003C8 17BC 0008 0018                         move.b  #rcv_inl4,buff_state(d0)    ;give that buffer to l4
      786 000003CE 3233 0028                              move.w  rim_buf_offset(d0),d1
      787 000003D2 4872 1000                              pea     0(a2,d1.w) ;pic_map(d1.w)
      788 000003D6 4EB9 0000 0000 E                       jsr     .l4_rcvintr               ;l4_rcvintr (faraddr)
      789 000003DC 588F                                   add.l   #4,sp
      790                                               endi
      791
      792                                             endi    ; RI interrupt
      793
```

```
ERR  LINE  ADDR
      795
      796                                ;
      797                                ;               Transmit interrupt processing
      798                                ;
      799
      800 000003DE 122A 3800                            move.b  rim_stat,d1             ;which interrupt bits
      801 000003E2 C22B 0017                            and.b   rim_int_copy,d1         ;mask says which are enabled
      802 000003E6 0201 0001                            and.b   #ta,d1
      803                                               if <ne> then.l
      804                                                                               ;------ TA interrupt -----
      805
      806 000003EE 52B9 0000 0000 E                       add.l   #1,.l4cnt_l2int_ta    ;count TA interrupt
      807 000003F4 177C 0000 0014                         move.b  #0,xmit_timer         ;cancel transmit timer
      808
      809 000003FA 303C 0000                              move    #buf0,d0                          ;get index to which buffer it was
      810 000003FE 0C33 0004 0018                         cmp.b   #xmit_enabled,buff_state(d0)      ;in bx
      811                                                 if <ne> then.s
      812 00000406 303C 0002                                move  #buf1,d0
      813                                                 endi
      814
      815 0000040A 122A 3800                              move.b  rim_stat,d1
      816 0000040E 0201 0002                              and.b   #tma,d1               ;TA without TMA?
      817                                                 if <eq> then.s
      818 00000414 52B9 0000 0000 E                         add.l  #1,.l4cnt_xmitnoack          ;yes: increment count
      819 0000041A 0C2B 0001 0015                           cmp.b  #XMIT_RETRIES,n_xmit_retries ;are we allowed a retry?
      820                                                   if <lt> then.s
      821 00000422 52B9 0000 0000 E                           add.l   #1,.l4cnt_p_retries       ;yes: increment global count
      822 00000428 52AB 0015                                  add.l   #1,n_xmit_retries         ;and count for this packet
      823 0000042C 1233 0020                                  move.b  enable_cmds(d0),d1        ;enable (re)transmit
      824 00000430 1541 3801                                  move.b  d1,rim_cmd        ;enable transmit
      825 00000434 177C 000F 0014                             move.b  #TO_XMIT_PKT,xmit_timer   ;start transmit timer
      826 0000043A 4EEB 041C 4E71                             jmp     end_TA                    ;wait for next interrupt
      827                                                   endi
      828                                                 endi
      829
      830                                ;                We are done with the buffer
      831
      832 00000440 17BC 0001 0018                         move.b  #xmit_empty,buff_state(d0)    ;mark it empty
      833 00000446 0A40 0002                              eor     #other_buffer,d0              ;look at the other buffer
      834 0000044A 0C33 0003 0018                         cmp.b   #xmit_full,buff_state(d0)
      835                                                 if <eq> then.s                        ;if it's full,
      836 00000452 1233 0020                                move.b  enable_cmds(d0),d1
      837 00000456 1541 3801                                move.b  d1,rim_cmd                ;enable transmit
      838 0000045A 177C 000F 0014                           move.b  #TO_XMIT_PKT,xmit_timer   ;start transmit timer
      839 00000460 177C 0000 0015                           move.b  #0,n_xmit_retries         ;start retry counter
      840 00000466 17BC 0004 0018                           move.b  #xmit_enabled,buff_state(d0) ;"it is sending"
      841                                                 else.s                                ;if no buffers to xmit
      842 0000046E 022B 00FE 0017                           and.b   #255-ta,rim_int_copy
      843 00000474 122B 0017                                move.b  rim_int_copy,d1           ;disable transmit interrupts
      844 00000478 1541 3800                                move.b  d1,rim_int
```

```
ERR  LINE  ADDR
      845                                                endi
      846
      847 0000047C 0C2B 0000 0016                        cmp.b   #0,n_owed_buffs                ;do we owe L4 a buffer?
      848                                                if <ne> then.s
      849 00000484 0C33 0002 0018                          cmp.b  #xmit_in14,buff_state(d0)     ;and is the other transm
      850                                                  if <ne> then.s                       ;  not already given to
      851 0000048C 532B 0016                                 sub.b   #1,n_owed_buffs            ;yes: reduce the count
      852 00000490 0A40 0002                                 eor     #other_buffer,d0           ;switch back to empty buffer
      853 00000494 17BC 0002 0018                            move.b  #xmit_in14,buff_state(d0) ;give that buffer to 14
      854 0000049A 3233 0028                                 move.w  rim_buf_offset(d0),d1
      855 0000049E 4872 1000                                 pea     0(a2,d1.w) ;pic_map(d1.w)
      856 000004A2 4EB9 0000 0000 E                          jsr     .14_gotbuf                 ;14_gotbuf (addr)
      857 000004A8 588F                                      add.l   #4,sp
      858                                                  endi
      859                                                endi
      860                                              endi
      861
      862                              end_TA:
```

```
ERR  LINE  ADDR
      864
      865                                ;
      866                                ;               Interrupt cleanup and return
      867                                ;
      868
      869
      870 000004AA 122A 3800                            move.b  rim_stat,d1
      871 000004AE 0201 0014                            and.b   #recon+por,d1
      872                                               if <ne> then.s                  ;------ RECON interrupt -----
      873                                                                               ; (or POR, which should never occur)
      874
      875 000004B4 52B9 0000 0000 E                       add.l   #1,.l4cnt_l2int_recon         ;count recon interrupt
      876 000004BA 157C 001E 3801                         move.b  #rimcmd_clrflgs,rim_cmd       ;clear POR and RECON flags
      877
      878                                               endi
      879
      880                                ; Check that there is no pending interrupt now.  There might be if the RIM
      881                                ; caused an interrupt condition during this routine.  Since the 8259 interrupt
      882                                ; controller in the PC is programmed for edge-triggered mode, we would prevent
      883                                ; an edge and therefore miss an interrupt if, say, the RI interrupt was turned
      884                                ; on after we checked for it but before we reset the TA interrupt.  Note that
      885                                ; checking here may cause a false interrupt later (an interrupt for which no
      886                                ; RIM interrupt bits are set), but that is harmless.  Losing an interrupt is not
      887
      888 000004C0 122A 3800                            move.b  rim_stat,d1             ;which interrupt bits
      889 000004C4 C22B 0017                            and.b   rim_int_copy,d1         ;mask says which are enabled
      890 000004C8 0201 0095                            and.b   #ta+ri+recon+por,d1
      891 000004CC 6600 FE9A                            bne     check_interrupts;we got a fresh one
      892
      893
      894                                ; Note that interrupts have been disabled all this time, including the
      895                                ; time to call the C routines.  That is probably not necessary, and is
      896                                ; undesirable because the C routines can take quite a while as they copy
      897                                ; data and headers from the packet buffers.  Think long and hard about race
      898                                ; conditions before changing that, however!
      899
      900
      901                                      ;;;;      movem.l (sp)+,d0/d1/a0/a1/a2/a3       ;working registers
      902 000004D0 4CDF 0C00                            movem.l (sp)+,a2/a3
      903                                      ;;;;      rte
      904 000004D4 4E75                                 rts
      905
      906                                               endu    a2,a3
```

```
ERR  LINE  ADDR
      908                              ;-------------------------------------------------------------------------
      909                              ;
      910                              ;       Timer interrupt routine
      911                              ;
      912                              ; This is a periodic 220 msec interrupt.
      913                              ; We call the l2_timerint() and l4_timerint() routines.
      914                              ;
      915                              ; Only a0, a1, d0, and d1 are saved here.
      916                              ; We do NOT switch stacks, so the running stack must be large enough
      917                              ; to accomodate whatever the timer interrupt routines use.
      918                              ;
      919                              ;-------------------------------------------------------------------------
      920
      921                                                  xdef    .timer_interrupt
      922                              .timer_interrupt:
      923 000004D6 48E7 C0C0                               movem.l d0/d1/a0/a1,-(sp)
      924
      925 000004DA 4EBA 0010 4E71                          jsr     l2_timerint     ;call Level 2 interrupt rtn
      926 000004E0 4EB9 0000 0000 E                        jsr     .l4_timerint    ;call Level 4 interrupt rtn
      927
      928 000004E6 4CDF 0303                               movem.l (sp)+,d0/d1/a0/a1
      929 000004EA 4E73                                    rte
      930
      931
      932
      933                              ;-------------------------------------------------------------------------
      934                              ;
      935                              ;       l2_timerint();     Level 2 timer interrupt routine
      936                              ;
      937                              ; Check to see if a transmit command has taken too long.
      938                              ; If so, abort the transmit.  That will cause TA to come on
      939                              ; after the token is next received by this station, and then
      940                              ; a NIC interrupt will occur so that the next transmission can
      941                              ; be scheduled.
      942                              ;
      943                              ; Changes no registers.
      944                              ;
      945                              ;-------------------------------------------------------------------------
      946
      947                                                  xdef    l2_timerint     ;(for debugging only)
      948                              l2_timerint:
      949
      950 000004EC 0C39 0000 0000 R                        cmp.b   #0,xmit_timer   ;are we transmitting?
                  00A2
      951                                                  if <ne> then.s          ;yes
      952 000004F6 5379 0000 00A2 R                          sub   #1,xmit_timer   ;count down
      953                                                    if  <eq> then.s     ;timeout!
      954 000004FE 13FC 0001 000F                              move.b  #rimcmd_xmit_dis,rim_cmd;disable transmitter
                  3801
      955 00000506 52B9 0000 0000 E                            add.l   #1,.l4cnt_xmittimeout   ;count transmit timeout
```

```
ERR  LINE  ADDR
      956                                                    ; we could call l4_trace to make a trace entry here,
      957                                                endi
      958                                            endi
      959 0000050C 4E75                              rts
      960
```

```
ERR  LINE  ADDR
      962
      963                              ;--------------------------------------------------------------------------
      964                              ;
      965                              ;       Long-pointer memory move routine for C.
      966                              ;
      967                              ;
      968                              ; movel (from, to, length)
      969                              ;
      970                              ;   addr from, to;
      971                              ;   int length;
      972                              ;
      973                              ;
      974                              ;  After our prologue, the stack looks like this:
      975                              ;
      976                              ;       12  length
      977                              ;        8  to
      978                              ;        4  from
      979                              ; sp--> 0  return address
      980                              ;
      981                              ; **** This is a quick-and-dirty.  We could optimize a lot.  ****
      982                              ;
      983                              ;--------------------------------------------------------------------------
      984
      985                                              xdef    .movel
      986                              .movel:
      987 0000050E 206F 0004                          move.l  4(sp),a0        ;from
      988 00000512 226F 0008                          move.l  8(sp),a1        ;to
      989 00000516 202F 000C                          move.l  12(sp),d0       ;length
      990 0000051A 5340                               sub     #1,d0
      991 0000051C 12D8                movel_loop:    move.b  (a0)+,(a1)+
      992 0000051E 51C8 FFFC                          dbra    d0,movel_loop
      993 00000522 4E75                               rts
      994
      995 00000524                                    end

      0 Errors,     0 Warnings
```

SYMBOL TABLE

ARC_ADDRESS		000F3802	BROADCAST_OK		00000000	BUF0		00000000
BUF1		00000002	BUF2		00000004	BUF3		00000006
BUFF_STATE	R ????14:	000000A6	CHECK_INTERRUPTS	R ????14:	00000368	DISABLE		00002700
ENABLE_CMDS	R ????14:	000000AE	END_TA	R ????14:	000004AA	ERROR	R ????14:	0000015A
INIT_EXIT	R ????14:	00000166	INIT_OK	R ????14:	00000162	INT_COND		000F0800
INT_VECTORS		00000060	L2_TIMERINT	R ????14:	000004EC	LOCAL_DATA	R ????14:	0000008E
MOVEL_LOOP	R ????14:	0000051C	NARG		00000000	NESTAR	R ????14:	0000008E
NIC_CTL		000F0801	NIC_STAT		000F0801	N_OWED_BUFFS	R ????14:	000000A4
N_XMIT_RETRIES	R ????14:	000000A3	OTHER_BUFFER		00000002	PAGE0		000F3000
PAGE1		000F3200	PAGE2		000F3400	PAGE3		000F3600
PANIC1	R ????14:	00000000	PANIC2	R ????14:	00000019	PANIC3	R ????14:	00000033
PANIC4	R ????14:	0000004F	PANIC5	R ????14:	0000006E	PICINTSV	R ????14:	0000009A
PIC_CHECK	R ????14:	000000FA	PIC_INTERRUPT	R ????14:	0000035A	PIC_INT_LEVEL		00000003
PIC_LOCATION		000F0000	PIC_MAP		000F0000	POR		00000010
PROM		000F0000	PROM_NW_ADDR		000F0007	RCV_EMPTY		00000005
RCV_ENABLED		00000006	RCV_FULL		00000007	RCV_INL4		00000008
RECON		00000004	RI		00000080	RIMCMD_CLRFLGS		0000001E
RIMCMD_CONFIG_L		0000000D	RIMCMD_CONFIG_S		00000005	RIMCMD_RECV_EN		00000004
RIMCMD_XMIT_DIS		00000001	RIMCMD_XMIT_EN		00000003	RIM_BUF_OFFSET	R ????14:	000000B6
RIM_CMD		000F3801	RIM_INT		000F3800	RIM_INTERRUPT		00000080
RIM_INT_COPY	R ????14:	000000A5	RIM_INT_ENB		00000001	RIM_STAT		000F3800
TA		00000001	TIMINTSV	R ????14:	0000009E	TMA		00000002
TO_XMIT_PKT		0000000F	XMIT_EMPTY		00000001	XMIT_ENABLED		00000004
XMIT_FULL		00000003	XMIT_INL4		00000002	XMIT_RETRIES		00000001
XMIT_TIMER	R ????14:	000000A2	XNSADDR	R ????14:	00000094	.L2_BUFF_STATE	R ????14:	000000A6
.L2_GETBUF	R ????14:	0000018C	.L2_INIT	R ????14:	000000BE	.L2_RCVRELEASE	R ????14:	000002A4
.L2_SENDBUF	R ????14:	000001DA	.L2_TERMINATE	R ????14:	0000016C	.L4CNT_L2INT_RECON	E	00000000
.L4CNT_L2INT_RI	E	00000000	.L4CNT_L2INT_TA	E	00000000	.L4CNT_P_RETRIES	E	00000000
.L4CNT_XMITNOACK	E	00000000	.L4CNT_XMITTIMEOUT	E	00000000	.L4_GOTBUF	E	00000000
.L4_RCVINTR	E	00000000	.L4_TIMERINT	E	00000000	.MOVEL	R ????14:	0000050E
.N_SERVICE	R ????14:	0000035A	.PANIC	E	00000000	.TIMER_INTERRUPT	R ????14:	000004D6