```
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
```

Wed 19-Mar-1986 18:43:50

Print request number 519

Station: $36

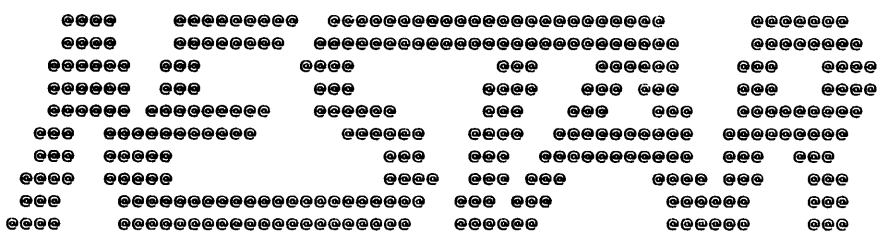
Name: L J Shustek

File Server: BUTLER ($FE)

NFS Pathname:

Filename (s):

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



```
;p tab 1,9,17,25,33,41,49,57,65,73
                title   Interrupt-driven Level
2         NESTAR CONFIDENTIAL
                page    58,132
;-------------------------------------------------------------------------------
;
;file: l2.asm
;
;
;   This is an 8088 version of interrupt-driven XNS Level 2 routines,
;   written primarily for debugging the Level 4 code in C.
;
;
;       The following assumptions are made:
;
;       1.  The C compiler is Microsoft version 3.0 in the small model,
;           except that certain pointers are declared to be FAR as
;           indicated.
;
;       2.  The NIC is located at segment address D200.
;
;       3.  There is an operational PC L4 running, from whose public data
;           area we can get our arcnet station address and XNS host address.
;
;       4.  There is no problem with taking over the use of the RIM from
;           the existing Level 4 and whatever other accesory programs are
;           loaded and running.  We do set the existing Level 4 "in use"
;           flag to encourage other processes to be idle.
;
;       5.  Registers ax, bx, cx and dx may be destroyed when called
;           from C routines.
;
;-------------------------------------------------------------------------------

;       (C) Copyright 1985, Nestar Systems, Inc.

;-------------------------------------------------------------------------------
;
;       Change log
;
;  12/ 2/85  L. Shustek   Written for the prototype fileserver L4.
;  12/ 6/85  L. Shustek   Add l2_terminate().
;  12/15/85  L. Shustek   Fix iret from timer interrupt routine.
;  12/19/85  L. Shustek   Reenable rcv interrupts from l2_rcvrelease().
;   1/24/86  L. Shustek   Disable interrupts in l2_getbuf().
;                         Fix group addressability for C data segment.
;   1/28/86  L. Shustek   Enable recon interrupts, and count them.
;   1/29/86  L. Shustek   Save es in c_from_int; clobbered if multiple interrupts!
;   2/ 2/86  L. Shustek   Switch to a fresh stack during interrupt processing.
;   2/ 4/86  L. Shustek   Initialize data so we can restart from the top.
;   2/ 6/86  L. Shustek   Fix so that no more than one empty transmit buffer
;                         at a time is given to L4.
;   2/11/86  L. Shustek   Add transmit retries for TA without TMA;  this raises
;                         the chance for successful error recovery.  See the
```

```
;                                       commentary in l4private.h about error strategy.
;   2/13/86  L. Shustek     Fix interrupt logic to guarantee an edge on the
;                           interrupt line.
;   2/18/86  L. Shustek     Move general routines into execasm.asm.
;   2/21/86  L. Shustek     Disallow broadcast reception temporarily, until
;                           supported by L4.
;   2/22/86  L. Shustek     Add l2_reverse_xns() to speed up packet processing.
;   2/25/86  L. Shustek     Have l2_reverse_xns reverse allno too.
;   3/19/86  L. Shustek     Verify NIC presence in l2_init by checking for ROM.
;
;-----------------------------------------------------------------------------

                .sall                   ;supress macro expansions

;;              include m:struct.mac            ;structure macros (not listed)
                .list
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;               General Symbols
                        ;
                        ;-----------------------------------------------------------------------

= 0004                  prescaleequ     4       ;timer prescale count (in 55 msec increments): 220 msec

= 000F                  TO_XMIT_PKT     equ     15      ; 14-15 * 220 msec = 3 sec packet transmit timeout
                                                        ; (Also change same symbol in l4private.h!)

= 0001                  XMIT_RETRIES    equ     1       ; how many transmit retries if TA without TMA
                                                        ; (Also change same symbol in l4private.h!)


= D200                  nic_location    equ     0d200h  ;default location for NIC


= 0013                  diskio          equ     13h     ;disk I/O interrupt - also L4 extended fcts

= 0021                  dos_int         equ     21h     ;dos function call interrupt; code in ah:
= 0025                  setvector       equ     25h     ;  set interrupt vector al from ds:dx
= 0035                  getvector       equ     35h     ;  get interrupt vector al into es:bx

= 001C                  timer_int       equ     1ch     ;timer interrupt user exit routine

= 0061                  spkr_port       equ     61h     ;speaker I/O port
= 0002                  spkr_enable     equ     2       ;enable bit in spkr_port
= 0001                  timer_gate      equ     1       ;timer channel two gate bit

= 0008                  l4_in_use       equ     08h     ;offset of l2_in_use in l4_publics
= 0009                  l4_long_mode    equ     09h     ;offset of long_pkt_mode flag in l4_publics
= 000A                  l4_our_ether    equ     0ah     ;offset of xns host address in l4_publics

= 0020                  irqcmd          equ     20h     ;8259 command register
= 0021                  irqmask         equ     21h     ;8259 mask register
= 0020                  eoi             equ     20h     ;end-of-interrupt command to 8259
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;               Network Interface Card symbols
                        ;
                        ;-----------------------------------------------------------------------

                        ;
                        ; RIM command register ("rim_cmd")
                        ;

= 0000                  broadcast_ok    equ     0       ;do we allow broadcast reception?  0=no, 1=yes

= 001E                  rimcmd_clrflgs  equ     1eh     ;RIM command: clear POR and RECON flags
= 000D                  rimcmd_config_l equ     0dh     ;RIM command: configure long packet mode
= 0005                  rimcmd_config_s equ     05h     ;RIM command: configure short packet mode
= 0004                  rimcmd_recv_en  equ     04h + 80h * broadcast_ok
                                                        ;RIM command: receive enable, add 8*buffer#
= 0003                  rimcmd_xmit_en  equ     03h     ;RIM command: transmit enable, add 8*buffer#
= 0001                  rimcmd_xmit_dis equ     01h     ;RIM command: disable transmit


                        ;
                        ; RIM status register ("rim_stat") and interrupt mask register ("rim_int")
                        ;

= 0080                  ri              equ     80h     ;RIM status: receiver inhibited
= 0010                  por             equ     10h     ;RIM status: power-on reset occurred
= 0004                  recon           equ     04h     ;RIM status: recon occurred
= 0002                  tma             equ     02h     ;RIM status: transmit message acknowledged
= 0001                  ta              equ     01h     ;RIM status: transmitter available


                        ;
                        ; NIC control register ("nic_ctl")
                        ;

= 0004                  rim_int_enb     equ     04h     ;enable RIM interrupts


                        ;
                        ; NIC status register ("nic_stat")
                        ;

= 0008                  rim_interrupt   equ     08h     ;RIM is interrupting
```

```
                                page

                        ;
                        ;       Packet buffer states
                        ;
                        ; Of the four Arcnet RIM buffers, the first two are used as
                        ; transmit buffers and the second two as receive buffers.
                        ; Each proceeds through the appropriate four states in sequence.
                        ; Buffer data structures are two bytes wide and indexed by 0,2,4,6.
                        ; We use the trick of XORing the buffer index to reference the "other"
                        ; transmit or receive buffer.
                        ;

= 0000                  buf0            equ     0       ;xmit buffer     ;buffer indices
= 0002                  buf1            equ     2       ;xmit buffer
= 0004                  buf2            equ     4       ;rcv buffer
= 0006                  buf3            equ     6       ;rcv buffer

= 0002                  other_buffer    equ     2       ;"other rcv/xmit buffer" XOR symbol

= 0001                  xmit_empty      equ     1       ;transmit buffer empty
= 0002                  xmit_in14       equ     2       ;transmit buffer given to Level 4 to fill
= 0003                  xmit_full       equ     3       ;transmit buffer full, awaiting transmission
= 0004                  xmit_enabled    equ     4       ;transmit buffer full, enabled for transmission

= 0005                  rcv_empty       equ     5       ;receive buffer empty
= 0006                  rcv_enabled     equ     6       ;receive buffer empty, enabled for reception
= 0007                  rcv_fullequ     7       ;receive buffer full, awaiting processing
= 0008                  rcv_in14equ     8       ;receive buffer given to Level 4 to empty
```

```
                                        page

                                ;
                                ;       Map of the NIC
                                ;


0000                            nic_map         segment at nic_location
0000  0200 [                    page0           db      512 dup (?)     ;buffer 0
            ??
                ]

0200  0200 [                    page1           db      512 dup (?)     ;buffer 1
            ??
                ]

0400  0200 [                    page2           db      512 dup (?)     ;buffer 2
            ??
                ]

0600  0200 [                    page3           db      512 dup (?)     ;buffer 3
            ??
                ]

0800  ??                        rim_int         db      ?               ; RIM interrupt mask register (write)
=                               rim_statequ     rim_int         ; RIM status register (read)
0801  ??                        rim_cmd         db      ?               ; RIM command register (write)
0802  ??                        nic_ctl         db      ?               ;NIC control register (write)
=                               nic_statequ     nic_ctl         ;NIC status register (read)
0803  ??                        nic_int         db      ?               ;NIC interrupt level register (read)
1000                                            org     1000h
1000  ????                      nic_rom         dw      ?               ;NIC rom
= AA55                          nic_signature   equ     0aa55h          ;NIC signature in first word
1002                            nic_map         ends
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;               DATA
                        ;
                        ;-----------------------------------------------------------------------


                        ;
                        ;               Global static data segment
                        ;               --------------------------
                        ;

                        dgroup          group   _data

0000                    _data           segment word public 'data'

                                        extrn   _l4cnt_l2int_ri:word    ;count of RI interrupts
                                        extrn   _l4cnt_l2int_ta:word    ;count of TA interrupts
                                        extrn   _l4cnt_l2int_recon:word ;count of recon interrupts
                                        extrn   _l4cnt_xmitnoack:word   ;count of TA w/o TMA
                                        extrn   _l4cnt_xmittimeout:word ;count of transmit timeouts
                                        extrn   _l4cnt_p_retries:word   ;count of packet xmit retries

0000  012C [                    int_stack_beg   db      300 dup (33h)   ;interrupt stack
              33
                 ]

= 012B                          int_stack       equ     $-1             ; (in data segement because cs=ss)

                        ;               Panic error messages
                        ;
                        ;  (Must be in the data segment because they are passed to C.)

012C  6C 32 5F 73 65 6E         panic1          db      "l2_sendbuf: bad buf addr",0
      64 62 75 66 3A 20
      62 61 64 20 62 75
      66 20 61 64 64 72
      00
0145  6C 32 5F 73 65 6E         panic2          db      "l2_sendbuf: bad buf state",0
      64 62 75 66 3A 20
      62 61 64 20 62 75
      66 20 73 74 61 74
      65 00
015F  6C 32 5F 72 63 76         panic3          db      "l2_rcvrelease: bad buf addr",0
      72 65 6C 65 61 73
      65 3A 20 62 61 64
      20 62 75 66 20 61
      64 64 72 00
017B  6C 32 5F 72 63 76         panic4          db      "l2_rcvrelease: bad buf state 1",0
```

```
      72 65 6C 65 61 73
      65 3A 20 62 61 64
      20 62 75 66 20 73
      74 61 74 65 20 31
      00
019A  6C 32 5F 72 63 76       panic5          db      "l2_rcvrelease: bad buf state 2",0
      72 65 6C 65 61 73
      65 3A 20 62 61 64
      20 62 75 66 20 73
      74 61 74 65 20 32
      00

01B9                          _data           ends


                              ;
                              ;                       Code segment data
                              ;                       -----------------
                              ;
                              ;               (None of this is accesible to C.)
                              ;

0000                          _text           segment byte public 'code'

                                              assume  cs:_text

                                              extrn   _l4_timerint:near       ;Level 4 timer interrupt routine
                                              extrn   _l4_rcvintr:near        ;Level 4 receive interrupt routine
                                              extrn   _l4_gotbuf:near         ;Level 4 transmit buffer avail int rtn
                                              extrn   _panic:near             ;failed assertion routine

                                              public  l2_data
0000                          l2_data         proc    near

0000  D200                    nic_seg         dw      nic_location    ;NIC segment address
0002  0000 0000 0000          xnsaddr         dw      0,0,0           ;our 6-byte XNS address
0008  00 00 00 00             l4_publics      dd      0               ;pointer to existing L4 publics
000C  ??                      nic_int_level   db      ?               ;nic interrupt level
000D  ????????                nicintsvdd      ?               ;previous nic interrupt vector
0011  ????????                timintsvdd      ?               ;previous timer interrupt vector
0015  ??                      intmasksv       db      ?               ;previous 8259 mask
0016  ??                      spkrsv          db      ?               ;previous speaker state
0017  04                      timer_prescale  db      prescale        ;timer prescale counter
0018  00                      xmit_timer      db      0               ;transmit packet timer
0019  00                      n_xmit_retries  db      0               ;transmit retry count
001A  00                      n_owed_buffs    db      0               ;number of buffers we owe to l4
001B  00                      rim_int_copy    db      0               ;copy of rim_int, the interrupt mask
001C  ????                    c_dseg          dw      ?               ;C's ds register
001E  ????                    int_ss          dw      ?               ;stack segment when interrupt occurs
0020  ????                    int_sp          dw      ?               ;stack pointer when interrupt occurs
```

```
                                        page

                                ;               Buffer control

= 0022                          _l2_buff_state  equ     this byte       ;external defn for debugging print
                                                public  _l2_buff_state

0022  01 00                     buff_state      db      xmit_empty,0    ;transmit buffer #0 starts empty
0024  01 00                                     db      xmit_empty,0    ;transmit buffer #1 starts empty
0026  05 00                                     db      rcv_empty,0     ;receive buffer #2 starts empty
0028  05 00                                     db      rcv_empty,0     ;receive buffer #3 starts empty

002A  03 00                     enable_cmds     db      rimcmd_xmit_en+8*0,0    ;buffer xmit or rcv enable cmds
002C  0B 00                                     db      rimcmd_xmit_en+8*1,0
002E  14 00                                     db      rimcmd_recv_en+8*2,0
0030  1C 00                                     db      rimcmd_recv_en+8*3,0

0032  0000                      rim_buf_offset  dw      page0-nic_map           ;offsets to rim buffers
0034  0200                                     dw      page1-nic_map
0036  0400                                     dw      page2-nic_map
0038  0600                                     dw      page3-nic_map

003A                            l2_data         endp



                                ;
                                ;               Macro to click the speaker
                                ;
                                ; Destroys al.
                                ;

                                click           macro
                                                in      al,spkr_port            ;current status
                                                xor     al,spkr_enable          ;reverse speaker bit
                                                and     al,not timer_gate       ;
                                                out     spkr_port,al            ;new status
                                                endm
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       boolean _l2_init ( &our_addr )
                        ;
                        ;               short int our_addr[3]
                        ;
                        ;
                        ;       Initialize Level 2 and return the 6-byte host station address.
                        ;       Return TRUE if initialization succeeded.
                        ;       (If the address is returned as 0, there is no existing Level 4.)
                        ;
                        ;-----------------------------------------------------------------------

                                public  _l2_init
003A                    _l2_initproc    near
003A  55                                push    bp
003B  8B EC                             mov     bp,sp
003D  57                                push    di
003E  56                                push    si

003F  1E                                push    ds
0040  06                                push    es

0041  2E: 8C 1E 001C R                  mov     c_dseg,ds               ;save c's ds

                        ;
                        ;               Miscellaneous data re-initialization
                        ;

0046  2E: C6 06 001A R 00               mov     n_owed_buffs,0                  ;we owe no buffers

004C  2E: C6 06 0022 R 01               mov     buff_state+buf0,xmit_empty      ;transmit buffer #0 starts empty
0052  2E: C6 06 0024 R 01               mov     buff_state+buf1,xmit_empty      ;transmit buffer #1 starts empty
0058  2E: C6 06 0026 R 05               mov     buff_state+buf2,rcv_empty       ;receive buffer #2 starts empty
005E  2E: C6 06 0028 R 05               mov     buff_state+buf3,rcv_empty       ;receive buffer #3 starts empty


                        ;
                        ;               Check if the NIC is really there
                        ;

0064  2E: A1 0000 R                     mov     ax,nic_seg              ;address the nic
0068  8E D8                             mov     ds,ax
                                        assume  ds:nic_map
006A  81 3E 1000 R AA55                 cmp     nic_rom,nic_signature   ;check for rom signature
                                        $ifnot  e
0072  E9 013C R                           jmp   error
                                        $endif
                                        assume  ds:nothing


                        ;
```

```
                                ;               Save existing Level 4 info and get our host address
                                ;

0075  B8 FFFF                                   mov     ax,0ffffh               ;Find existing L4
0078  CD 13                                     int     diskio                  ; &publics in ds:di
                                                $if     c
007C  E9 013C R                                   jmp   error                   ; No existing L4!
                                                $endif

                                ; To work if there is no existing L4 in the machine, we would have to do a
                                ; soft reset to get the arcnet address (lsb of xns host address), and get
                                ; the other bytes of the host address from the rom on the nic.

007F  2E: 89 3E 0008 R                          mov     word ptr l4_publics,di  ;save &publics

0084  2E: 8C 1E 000A R                          mov     word ptr l4_publics+2,ds  ;move xns host address
0089  8B 45 0A                                  mov     ax,ds:[di+l4_our_ether+0]
008C  2E: A3 0002 R                             mov     xnsaddr+0,ax
0090  8B 45 0C                                  mov     ax,ds:[di+l4_our_ether+2]
0093  2E: A3 0004 R                             mov     xnsaddr+2,ax
0097  8B 45 0E                                  mov     ax,ds:[di+l4_our_ether+4]
009A  2E: A3 0006 R                             mov     xnsaddr+4,ax

009E  C6 45 08 01                               mov     byte ptr ds:[di+l4_in_use],1    ;set L4 busy flag

                                ;
                                ;               Initialize the NIC
                                ;

00A2  2E: A1 0000 R                             mov     ax,nic_seg              ;address the nic
00A6  8E D8                                     mov     ds,ax
                                                assume  ds:nic_map

00A8  C6 06 0801 R 1E                           mov     rim_cmd,rimcmd_clrflgs  ;clear POR and RECON flags
00AD  C6 06 0801 R 0D                           mov     rim_cmd,rimcmd_config_l ;configure for long packets
00B2  A0 0803 R                                 mov     al,nic_int              ;save interrupt level
00B5  24 07                                     and     al,7
00B7  2E: A2 000C R                             mov     nic_int_level,al

                                                assume  ds:nothing

                                ;
                                ;               Setup the NIC interrupt routine
                                ;

00BB  2E: A0 000C R                             mov     al,nic_int_level
00BF  04 08                                     add     al,8                    ;vector # = interrupt + 8
00C1  B4 35                                     mov     ah,getvector
00C3  CD 21                                     int     dos_int                 ;get old vector in es:bx
00C5  2E: 89 1E 000D R                          mov     word ptr nicintsv,bx    ;save it
00CA  2E: 8C 06 000F R                          mov     word ptr nicintsv+2,es

00CF  BA 0365 R                                 mov     dx,offset nic_interrupt ;&our rtn in ds:dx
00D2  8C C8                                     mov     ax,cs
```

```
00D4  8E D8                                   mov     ds,ax
00D6  2E: A0 000C R                           mov     al,nic_int_level
00DA  04 08                                   add     al,8                    ;vector # = interrupt + 8
00DC  B4 25                                   mov     ah,setvector            ;setup vector to our rtn
00DE  CD 21                                   int     dos_int

                                      ;
                                      ;       Setup the timer interrupt routine
                                      ;

00E0  B0 1C                                   mov     al,timer_int
00E2  B4 35                                   mov     ah,getvector
00E4  CD 21                                   int     dos_int                 ;get old vector in es:bx
00E6  2E: 89 1E 0011 R                        mov     word ptr timintsv,bx    ;save it
00EB  2E: 8C 06 0013 R                        mov     word ptr timintsv+2,es

00F0  BA 0500 R                               mov     dx,offset timer_interrupt
00F3  8C C8                                   mov     ax,cs                   ;&our routine in ds:dx
00F5  8E D8                                   mov     ds,ax
00F7  B0 1C                                   mov     al,timer_int
00F9  B4 25                                   mov     ah,setvector            ;set vector to our rtn
00FB  CD 21                                   int     dos_int

                                      ;
                                      ;       Set the 8259 interrupt controller to allow NIC interrupts
                                      ;

00FD  E4 21                                   in      al,irqmask              ;current mask
00FF  2E: A2 0015 R                           mov     intmasksv,al            ;save it
0103  2E: 8A 0E 000C R                        mov     cl,nic_int_level        ;compute nic interrupt mask
0108  B3 FE                                   mov     bl,0feh
010A  D2 C3                                   rol     bl,cl
010C  22 C3                                   and     al,bl                   ;combine with existing mask
010E  E6 21                                   out     irqmask,al              ;output new 8259 mask

0110  E4 61                                   in      al,spkr_port            ;save state of the speaker port
0112  2E: A2 0016 R                           mov     spkrsv,al

                                      ;
                                      ;       Enable NIC interrupts and start a receive command
                                      ;

0116  2E: A1 0000 R                           mov     ax,nic_seg              ;address the nic
011A  8E D8                                   mov     ds,ax
                                              assume  ds:nic_map

011C  C6 06 0802 R 04                         mov     nic_ctl,rim_int_enb     ;enable nic interrupts
0121  2E: A0 002E R                           mov     al,enable_cmds+buf2     ;enable rcv on buffer 2
0125  A2 0801 R                               mov     rim_cmd,al
0128  2E: C6 06 0026 R 06                     mov     buff_state+buf2,rcv_enabled     ;change its state
012E  C6 06 0800 R 80                         mov     rim_int,ri              ;enable rcv and recon interrupts
0133  2E: C6 06 001B R 84                     mov     rim_int_copy,ri+recon
0139  EB 06 90                                jmp     init_ok
```

```
                                        assume  ds:nothing

                                ;
                                ;               Return to the caller with the host address.
                                ;

013C  33 C0                     error:          xor     ax,ax           ;return ax = FALSE if error
013E  EB 04 90                                  jmp     init_exit

0141  B8 0001                   init_ok:        mov     ax,1            ;return ax = TRUE if no error


0144  07                        init_exit:      pop     es
0145  1F                                        pop     ds

0146  8B 5E 04                                 mov     bx,[bp+4]               ;&our_addr
0149  2E: 8B 0E 0002 R                         mov     cx,xnsaddr+0
014E  89 0F                                     mov     ds:[bx+0],cx
0150  2E: 8B 0E 0004 R                         mov     cx,xnsaddr+2
0155  89 4F 02                                 mov     ds:[bx+2],cx
0158  2E: 8B 0E 0006 R                         mov     cx,xnsaddr+4
015D  89 4F 04                                 mov     ds:[bx+4],cx

0160  5E                                        pop     si
0161  5F                                        pop     di
0162  5D                                        pop     bp
0163  C3                                        ret                             ;return with AX set
0164                            _12_initendp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       l2_terminate ()
                        ;
                        ;  Close down level 2.
                        ;  Remove all interrupt routines.
                        ;
                        ;-----------------------------------------------------------------------

                                public  _l2_terminate
0164                    _l2_terminate   proc    near
0164  55                                push    bp
0165  8B EC                            mov     bp,sp
0167  57                                push    di
0168  56                                push    si
0169  1E                                push    ds
016A  06                                push    es

016B  2E: A0 0015 R                     mov     al,intmasksv            ;restore original 8259 mask
016F  E6 21                             out     irqmask,al

0171  2E: A0 0016 R                     mov     al,spkrsv               ;restore speaker
0175  E6 61                             out     spkr_port,al

0177  2E: A1 0000 R                     mov     ax,nic_seg              ;address the nic
017B  8E D8                             mov     ds,ax
                                        assume  ds:nic_map
017D  C6 06 0802 R 00                   mov     nic_ctl,0       ;disable nic interrupts
0182  C6 06 0800 R 00                   mov     rim_int,0


0187  2E: A1 0008 R                     mov     ax,word ptr l4_publics  ;if there was an old Level 4
018B  2E: 0B 06 000A R                  or      ax,word ptr l4_publics+2
                                        $ifnot  z
0192  2E: 8B 3E 0008 R                   mov   di,word ptr l4_publics  ;clear Level 4 busy flag
0197  2E: 8E 06 000A R                   mov   es,word ptr l4_publics+2
019C  26: C6 45 08 00                    mov   byte ptr es:[di+l4_in_use],0
01A1  26: F6 45 09 01                    test  byte ptr es:[di+l4_long_mode],1 ;and reset long/short packet mode
                                         $if   z
01A8  C6 06 0801 R 05                      mov rim_cmd,rimcmd_config_s ;short packet mode
                                         $else
01B0  C6 06 0801 R 0D                      mov rim_cmd,rimcmd_config_l ;long packet mode
                                         $endif
                                         assume ds:nothing
                                        $endif

01B5  2E: 8B 16 000D R                  mov     dx,word ptr nicintsv    ;restore nic interrupt rtn
01BA  2E: 8E 1E 000F R                  mov     ds,word ptr nicintsv+2
01BF  2E: A0 000C R                     mov     al,nic_int_level
01C3  04 08                             add     al,8
01C5  B4 25                             mov     ah,setvector
01C7  CD 21                             int     dos_int
```

```
01C9  2E: 8B 16 0011 R                        mov     dx,word ptr timintsv    ;restore timer interrupt rtn
01CE  2E: 8E 1E 0013 R                        mov     ds,word ptr timintsv+2
01D3  B0 1C                                   mov     al,timer_int
01D5  B4 25                                   mov     ah,setvector
01D7  CD 21                                   int     dos_int

01D9  07                                      pop     es
01DA  1F                                      pop     ds
01DB  5E                                      pop     si
01DC  5F                                      pop     di
01DD  5D                                      pop     bp
01DE  C3                                      ret
01DF                            _12_terminate endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       faraddr _l2_getbuf ()
                        ;
                        ; Get an empty transmit buffer and return a long pointer to it.
                        ;
                        ; If there is no free buffer, return NIL and sometime later call
                        ; l4_gotbuf( faraddr ) as an interrupt routine and provide it the
                        ; long pointer to the buffer.  Only one call to l4_gotbuf() should
                        ; be outstanding until the buffer is freed with a call to l2_sendbuf().
                        ;
                        ;  Arcnet logic:
                        ;
                        ;       IF buffer_0 is empty, return buffer_0;
                        ;       IF buffer_1 is empty, return buffer_1;
                        ;       OTHERWISE  ++ n_owed_buffs;
                        ;                  return NIL;
                        ;
                        ;-----------------------------------------------------------------------

                                public  _l2_getbuf
01DF                    _l2_getbuf      proc    near
01DF  55                                push    bp
01E0  8B EC                             mov     bp,sp

01E2  9C                                pushf           ;-------
01E3  FA                                cli             ;------- disable interrupts

01E4  2E: 80 3E 0022 R 01               cmp     buff_state+buf0,xmit_empty      ;is buffer 0 available?
                                        $if     e
01EC  2E: C6 06 0022 R 02                 mov   buff_state+buf0,xmit_in14       ;yes: use it
01F2  B8 0000                             mov   ax,page0-nic_map
01F5  2E: 8B 16 0000 R                    mov   dx,nic_seg
                                        $else

01FD  2E: 80 3E 0024 R 01               cmp     buff_state+buf1,xmit_empty      ;is buffer 1 available?
                                        $if     e
0205  2E: C6 06 0024 R 02                 mov   buff_state+buf1,xmit_in14       ;yes: use it
020B  B8 0200                             mov   ax,page1-nic_map
020E  2E: 8B 16 0000 R                    mov   dx,nic_seg
                                        $else

0216  2E: FE 06 001A R                  inc     n_owed_buffs                    ;no buffer: remember request

021B  33 C0                             xor     ax,ax
021D  33 D2                             xor     dx,dx                           ;and return NIL

                                        $endif
                                        $endif

021F  9D                                popf            ;------ restore interrupts
```

```
 0220  5D                                      pop     bp
 0221  C3                                      ret
 0222                           _12_getbuf     endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       l2_sendbuf ( faraddr buffer)
                        ;
                        ; Send, or queue for sending, the transmit buffer whose address is
                        ; supplied.  This must be a buffer previously supplied by l2_getbuf()
                        ; or to l4_gotbuf();
                        ;
                        ; If we have an empty buffer and still owe one to L4, give it him now.
                        ; This is because L4 can only handle one empty transmit buffer at a time.
                        ;
                        ; Arcnet logic:
                        ;       assert: buffer is buffer_0 or buffer_1
                        ;       assert: buffer status is "in L4".
                        ;       IF the status of the other buffer is "transmitting"
                        ;         THEN  set buffer status to "awaiting transmit"
                        ;         ELSE  start transmitting this buffer;
                        ;               set status to "transmitting";
                        ;               IF n_owed_buffs>0
                        ;               AND the status of the other buffer is "empty"
                        ;                 THEN --n_owed_buffer;
                        ;                      l4_gotbuf(the other buffer)
                        ;
                        ;-----------------------------------------------------------------------

                                public  _l2_sendbuf
0222                    _l2_sendbuf     proc    near
0222  55                                push    bp
0223  8B EC                            mov     bp,sp
0225  06                                push    es

0226  8B 46 04                         mov     ax,[bp+4]       ;get offset part of faraddr
0229  3D 0000                          cmp     ax,page0-nic_map
                                        $if     e
022E  BB 0000                             mov   bx,buf0         ;it is buffer 0
                                        $else
0234  3D 0200                           cmp     ax,page1-nic_map
                                        $if     e
0239  BB 0002                             mov   bx,buf1         ;it is buffer 1
                                        $else
023F  B8 012C R                           mov   ax,offset dgroup:panic1 ;neither: _panic("bad buf addr")
0242  50                                  push  ax
0243  E8 0000 E                           call  _panic
                                        $endif
                                        $endif

0246  2E: 80 BF 0022 R 02               cmp     buff_state[bx],xmit_in14;assert buffer was in 14
                                        $ifnot  e
024E  B8 0145 R                           mov   ax,offset dgroup:panic2
0251  50                                  push  ax
0252  E8 0000 E                           call  _panic          ;panic("bad buf state")
                                        $endif
```

```
0255  9C                                    pushf           ;-------
0256  FA                                    cli             ;------- disable interrupts

0257  81 F3 0002                           xor     bx,other_buffer                 ;look at the other buffer
025B  2E: 80 BF 0022 R 04                  cmp     buff_state[bx],xmit_enabled     ;sending?

                                           $if     e                               ;yes: xmitter is busy
0263  81 F3 0002                             xor   bx,other_buffer                 ;switch back to our buffer
0267  2E: C6 87 0022 R 03                    mov   buff_state[bx],xmit_full        ;state is "awaiting xmit"

                                           $else                                   ;xmitter is free
0270  81 F3 0002                             xor   bx,other_buffer                 ;switch back to our buffer
0274  2E: 8A 87 002A R                       mov   al,enable_cmds[bx]
0279  2E: 8E 06 0000 R                       mov   es,nic_seg
027E  26: A2 0801 R                          mov   es:rim_cmd,al                   ;enable transmit
0282  2E: C6 06 0018 R 0F                    mov   xmit_timer,TO_XMIT_PKT          ;start transmit timer
0288  2E: C6 06 0019 R 00                    mov   n_xmit_retries,0                ;start retry counter
028E  2E: C6 87 0022 R 04                    mov   buff_state[bx],xmit_enabled     ;"we are sending"
0294  2E: 80 0E 001B R 01                    or    rim_int_copy,ta
029A  2E: A0 001B R                          mov   al,rim_int_copy                 ;enable transmit interrupts
029E  26: A2 0800 R                          mov   es:rim_int,al

02A2  2E: 80 3E 001A R 00                    cmp   n_owed_buffs,0                  ;if we owe buffers
                                             $if   a
02AA  81 F3 0002                               xor bx,other_buffer
02AE  2E: 80 BF 0022 R 01                      cmp buff_state[bx],xmit_empty       ; and the other buffer is empty
                                               $if e
02B6  2E: FE 0E 001A R                             dec  n_owed_buffs               ; then reduce the count owed
02BB  2E: C6 87 0022 R 02                          mov  buff_state[bx],xmit_in14   ; and give that buffer to 14
02C1  B8 0000 E                                    mov  ax,offset _14_gotbuf
02C4  E8 04DE R                                    call c_from_int                 ; 14_gotbuf (faraddr)
                                               $endif
                                            $endif
                                           $endif

02C7  9D                                   popf            ;------ restore interrupts

02C8  07                                   pop     es
02C9  5D                                   pop     bp
02CA  C3                                   ret
02CB                          _12_sendbuf  endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       l2_rcvrelease ( faraddr buffer )
                        ;
                        ;  Release the received packet whose buffer address is supplied.
                        ;  It was previously provided by a call to l4_rcvintr().
                        ;  This can cause l4_rcvintr() to be called if another packet is ready.
                        ;
                        ;  Arcnet logic:
                        ;       assert: buffer is buffer_2 or buffer_3
                        ;       assert: buffer status was "in L4"
                        ;       IF the other buffer is not receiving
                        ;         THEN  enable receive on this buffer
                        ;               assert: other buffer is full of data
                        ;               l4_rcvintr (other buffer)
                        ;
                        ;-----------------------------------------------------------------------

                                public  _l2_rcvrelease
02CB                    _l2_rcvrelease  proc    near
02CB  55                                push    bp
02CC  8B EC                             mov     bp,sp

02CE  8B 46 04                          mov     ax,[bp+4]       ;get offset part of faraddr
02D1  3D 0400                           cmp     ax,page2-nic_map
                                        $if     e
02D6  BB 0004                             mov   bx,buf2         ;it is buffer 2
                                        $else
02DC  3D 0600                           cmp     ax,page3-nic_map
                                        $if     e
02E1  BB 0006                             mov   bx,buf3         ;it is buffer 3
                                        $else
02E7  B8 015F R                           mov   ax,offset dgroup:panic3 ;neither: panic("bad buf addr")
02EA  50                                  push  ax
02EB  E8 0000 E                           call  _panic
                                        $endif
                                        $endif

02EE  2E: 80 BF 0022 R 08               cmp     buff_state[bx],rcv_inl4 ;assert buffer was in l4
                                        $ifnot  e
02F6  B8 017B R                           mov   ax,offset dgroup:panic4
02F9  50                                  push  ax
02FA  E8 0000 E                           call  _panic          ;panic("bad buf state")
                                        $endif

02FD  9C                                pushf                   ;-------
02FE  FA                                cli                     ;------- disable interrupts

02FF  2E: C6 87 0022 R 05               mov     buff_state[bx],rcv_empty;buffer is now empty

0305  81 F3 0002                        xor     bx,other_buffer         ;look at the other rcv buffer
0309  2E: 80 BF 0022 R 06               cmp     buff_state[bx],rcv_enabled
```

```
                                                $ifnot  e                       ;if it is not enabled for rcv
                                                                                ;then it must be full of data
 0311  06                                         push  es
 0312  81 F3 0002                                  xor bx,other_buffer           ;switch back to buffer just freed
 0316  2E: 8A 87 002A R                            mov al,enable_cmds[bx]
 031B  2E: 8E 06 0000 R                            mov es,nic_seg
 0320  26: A2 0801 R                               mov es:rim_cmd,al                     ;enable receive
 0324  2E: C6 87 0022 R 06                         mov buff_state[bx],rcv_enabled        ;"we are receiving"
 032A  2E: 80 0E 001B R 80                         or  rim_int_copy,ri                   ;enable rcv interrupts
 0330  2E: A0 001B R                               mov al,rim_int_copy
 0334  26: A2 0800 R                               mov es:rim_int,al
 0338  07                                         pop   es

 0339  81 F3 0002                                 xor   bx,other_buffer                  ;switch back to other rcv buffer
 033D  2E: 80 BF 0022 R 07                        cmp   buff_state[bx],rcv_full          ;assert it is full
                                                  $ifnot e
 0345  B8 019A R                                   mov  ax,offset dgroup:panic5
 0348  50                                          push ax
 0349  E8 0000 E                                   call _panic
                                                  $endif
 034C  2E: C6 87 0022 R 08                        mov   buff_state[bx],rcv_in14          ;give that buffer to 14
 0352  2E: FF 36 0000 R                           push  nic_seg
 0357  2E: FF B7 0032 R                           push  rim_buf_offset[bx]
 035C  E8 0000 E                                  call  _14_rcvintr                      ;14_rcvintr (faraddr)
 035F  83 C4 04                                   add   sp,4
                                                $endif

 0362  9D                                       popf                    ;------ restore interrupts
                                                                        ; (see the note in the interrupt routine.)

 0363  5D                                       pop     bp
 0364  C3                                       ret
 0365                           _12_rcvrelease  endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;
                        ;                       The NIC interrupt routine
                        ;
                        ;
                        ;
                        ;  This is a hardware interrupt routine, so NO registers can be changed,
                        ;  and NO assumptions can be made about their contents except for CS.
                        ;
                        ;
                        ;  If this is a "receive done" interrupt (RI just came on) then
                        ;       Mark the buffer as "full".
                        ;       If the other buffer is empty, enable receive on it.
                        ;       If the other buffer is not in L4, call l4_rcvintr and mark
                        ;          this buffer as "in l4".
                        ;
                        ;  If this is a "transmit done" interrupt (TA just came on) then
                        ;       Mark the buffer being transmitted as empty.
                        ;       If we got TA without TMA and we are eligible for a retry,
                        ;          restart transmission and exit.
                        ;       If the other buffer is full, start transmitting it,
                        ;          otherwise disable the transmitter interrupt.
                        ;       If we owe L4 a transmit buffer and the other buffer
                        ;       isn't already given to L4, call l4_gotbuf().
                        ;
                        ;  If this is a recon (or, impossibly, a power-on reset) interrupt,
                        ;  just clear the condition and increment a counter.
                        ;
                        ;  In high traffic situations it is fairly important to process receive
                        ;  interrupts before transmit interrupts.  If you do the reverse, and
                        ;  processing the transmit interrupt schedules more transmissions, the
                        ;  receive buffers can be backed up to the point where the other stations
                        ;  will timeout their transmission to us.
                        ;
                        ;-----------------------------------------------------------------------

                                        public  nic_interrupt   ;(for debugging only)
0365                    nic_interrupt   proc    near
0365  50                                push    ax
0366  53                                push    bx
0367  1E                                push    ds
0368  06                                push    es
                                        click                   ;click the speaker

0371  2E: 8E 1E 0000 R                  mov     ds,nic_seg      ;ds points to the nic
                                        assume  ds:nic_map
0376  2E: 8E 06 001C R                  mov     es,c_dseg       ;es points to C's data segment
                                        assume  es:dgroup

037B  2E: 8C 16 001E R                  mov     int_ss,ss       ;save stack segment
```

```
 0380  2E: 89 26 0020 R                     mov     int_sp,sp       ;save stack pointer
 0385  2E: 8E 16 001C R                     mov     ss,c_dseg       ;setup a new stack in our data segment
 038A  BC 012B R                            mov     sp,offset dgroup:int_stack
```

```
                                        page

 038D                           check_interrupts:

                                ;
                                ;               Receive interrupt processing
                                ;

 038D  A0 0800 R                                mov     al,rim_stat             ;which interrupt bits
 0390  2E: 22 06 001B R                         and     al,rim_int_copy         ;mask says which are enabled
 0395  A8 80                                    test    al,ri

                                                $ifnot  z                       ;------ RI interrupt -----

 0399  26: FF 06 0000 E                           inc   _14cnt_12int_ri         ;count RI interrupt

 039E  BB 0004                                    mov   bx,buf2                         ;get index to which buffer
 03A1  2E: 80 BF 0022 R 06                        cmp   buff_state[bx],rcv_enabled
                                                  $ifnot e
 03A9  BB 0006                                      mov bx,buf3
                                                  $endif
 03AC  2E: C6 87 0022 R 07                        mov   buff_state[bx],rcv_full         ;mark it full

 03B2  81 F3 0002                                 xor   bx,other_buffer                 ;look at the other rcv buffer
 03B6  2E: 80 BF 0022 R 05                        cmp   buff_state[bx],rcv_empty
                                                  $if   e                               ;if it is empty
 03BE  2E: 8A 87 002A R                             mov al,enable_cmds[bx]
 03C3  A2 0801 R                                    mov rim_cmd,al                     ;enable receive
 03C6  2E: C6 87 0022 R 06                          mov buff_state[bx],rcv_enabled     ;"it is receiving"
                                                   ;(rcv interrupts are already enabled.)
                                                  $else
 03CF  2E: 80 26 001B R 7F                          and rim_int_copy,255-ri
 03D5  2E: A0 001B R                                mov al,rim_int_copy                ;disable rcvr interrupts
 03D9  A2 0800 R                                    mov rim_int,al
                                                  $endif

 03DC  2E: 80 BF 0022 R 08                        cmp   buff_state[bx],rcv_in14         ;is the other buffer in 14?
                                                  $ifnot e                              ;if not,
 03E4  81 F3 0002                                   xor bx,other_buffer                 ;switch to the newly rcvd buffer
 03E8  2E: C6 87 0022 R 08                          mov buff_state[bx],rcv_in14         ;give that buffer to 14
 03EE  B8 0000 E                                    mov ax,offset _14_rcvintr
 03F1  E8 04DE R                                    call c_from_int                     ;14_rcvintr (faraddr)
                                                  $endif

                                                $endif  ; RI interrupt
```

```
                                        page

                                ;
                                ;       Transmit interrupt processing
                                ;

03F4  A0 0800 R                         mov     al,rim_stat             ;which interrupt bits
03F7  2E: 22 06 001B R                  and     al,rim_int_copy         ;mask says which are enabled
03FC  A8 01                             test    al,ta
                                        $if     z
0400  E9 04AB R                           jmp   end_TA
                                        $endif
                                                                        ;------ TA interrupt -----

0403  26: FF 06 0000 E                    inc   _l4cnt_l2int_ta         ;count TA interrupt
0408  2E: C6 06 0018 R 00                 mov   xmit_timer,0            ;cancel transmit timer

040E  BB 0000                             mov   bx,buf0                         ;get index to which buffer it was
0411  2E: 80 BF 0022 R 04                 cmp   buff_state[bx],xmit_enabled     ;in bx
                                          $ifnot e
0419  BB 0002                               mov bx,buf1
                                          $endif

041C  F6 06 0800 R 02                     test  rim_stat,tma            ;TA without TMA?
                                          $if   z
0423  26: FF 06 0000 E                      inc _l4cnt_xmitnoack                ;yes: increment count
0428  2E: 80 3E 0019 R 01                   cmp n_xmit_retries,XMIT_RETRIES     ;are we allowed a retry?
                                            $if b
0430  26: FF 06 0000 E                        inc  _l4cnt_p_retries             ;yes: increment global count
0435  2E: FE 06 0019 R                        inc  n_xmit_retries               ;and count for this packet
043A  2E: 8A 87 002A R                        mov  al,enable_cmds[bx]           ;enable (re)transmit
043F  A2 0801 R                               mov  rim_cmd,al                   ;enable transmit
0442  2E: C6 06 0018 R 0F                     mov  xmit_timer,TO_XMIT_PKT       ;start transmit timer
0448  EB 61                                   jmp  short end_TA                 ;wait for next interrupt
                                            $endif
                                          $endif

                                ;         We are done with the buffer

044A  2E: C6 87 0022 R 01                 mov   buff_state[bx],xmit_empty       ;mark it empty
0450  81 F3 0002                          xor   bx,other_buffer                 ;look at the other buffer
0454  2E: 80 BF 0022 R 03                 cmp   buff_state[bx],xmit_full
                                          $if   e                               ;if it's full,
045C  2E: 8A 87 002A R                      mov al,enable_cmds[bx]
0461  A2 0801 R                            mov rim_cmd,al                      ;enable transmit
0464  2E: C6 06 0018 R 0F                   mov xmit_timer,TO_XMIT_PKT          ;start transmit timer
046A  2E: C6 06 0019 R 00                   mov n_xmit_retries,0                ;start retry counter
0470  2E: C6 87 0022 R 04                   mov buff_state[bx],xmit_enabled     ;"it is sending"
                                          $else                                 ;if no buffers to xmit
0479  2E: 80 26 001B R FE                   and rim_int_copy,255-ta
047F  2E: A0 001B R                         mov al,rim_int_copy                 ;disable transmit interrupts
0483  A2 0800 R                            mov rim_int,al
                                          $endif
```

```
0486  2E: 80 3E 001A R 00                cmp   n_owed_buffs,0                ;do we owe L4 a buffer?
                                         $ifnot z
048E  2E: 80 BF 0022 R 02                  cmp  buff_state[bx],xmit_in14     ;and is the other transmit buffer
                                           $ifnot e
0496  2E: FE 0E 001A R                       dec  n_owed_buffs               ;yes: reduce the count
049B  81 F3 0002                             xor  bx,other_buffer            ;switch back to empty buffer
049F  2E: C6 87 0022 R 02                    mov  buff_state[bx],xmit_in14   ;give that buffer to L4
04A5  B8 0000 E                              mov  ax,offset _l4_gotbuf
04A8  E8 04DE R                              call c_from_int                 ;l4_gotbuf (faraddr)
                                           $endif
                                         $endif

04AB                           end_TA:
```

```
                                        page

                                ;
                                ;               Interrupt cleanup and return
                                ;


04AB  F6 06 0800 R 14                           test    rim_stat,recon+por
                                                $ifnot  z                       ;------ RECON interrupt -----
                                                                                ; (or POR, which should never occur)

04B2  26: FF 06 0000 E                            inc   _14cnt_12int_recon      ;count recon interrupt
04B7  C6 06 0801 R 1E                             mov   rim_cmd,rimcmd_clrflgs  ;clear POR and RECON flags

                                                $endif

                                ; Check that there is no pending interrupt now.  There might be if the RIM
                                ; caused an interrupt condition during this routine.  Since the 8259 interrupt
                                ; controller in the PC is programmed for edge-triggered mode, we would prevent
                                ; an edge and therefore miss an interrupt if, say, the RI interrupt was turned
                                ; on after we checked for it but before we reset the TA interrupt.  Note that
                                ; checking here may cause a false interrupt later (an interrupt for which no
                                ; RIM interrupt bits are set), but that is harmless.  Losing an interrupt is not.

04BC  A0 0800 R                                 mov     al,rim_stat             ;which interrupt bits
04BF  2E: 22 06 001B R                          and     al,rim_int_copy         ;mask says which are enabled
04C4  A8 95                                     test    al,ta+ri+recon+por
                                                $ifnot  z
04C8  E9 038D R                                   jmp   check_interrupts        ;we got a fresh one
                                                $endif


                                ; Note that interrupts have been disabled all this time, including the
                                ; time to call the C routines.  That is probably not necessary, and is
                                ; undesirable because the C routines can take quite a while as they copy
                                ; data and and from the packet buffers.  Think long and hard about race
                                ; conditions before changing that, however!

04C8  2E: 8E 16 001E R                          mov     ss,int_ss       ;restore stack segment
04D0  2E: 8B 26 0020 R                          mov     sp,int_sp       ;restore stack pointer

                                                assume  ds:nothing
                                                assume  es:nothing
04D5  B0 20                                     mov     al,eoi          ;clear 8259 interrupt
04D7  E6 20                                     out     irqcmd,al       ;to allow other interrupts
04D9  07                                        pop     es
04DA  1F                                        pop     ds
04DB  5B                                        pop     bx
04DC  58                                        pop     ax
04DD  CF                                        iret

04DE                            nic_interrupt   endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       Call a C routine from assembly language, perhaps from the NIC
                        ;       interrupt environment.
                        ;
                        ;       ax = routine to call with a pointer to the current buffer
                        ;            as an argument.
                        ;       bx = index of the current buffer.
                        ;
                        ;       Destroys  ax, bx
                        ;
                        ;-----------------------------------------------------------------------

04DE                    c_from_int      proc    near
04DE  1E                                push    ds              ;save stuff
04DF  06                                push    es
04E0  51                                push    cx
04E1  52                                push    dx              ;(C preserves si and di!)

04E2  2E: 8E 1E 001C R                  mov     ds,c_dseg       ;setup C's data segment
04E7  2E: 8E 06 001C R                  mov     es,c_dseg       ;setup C's data segment

04EC  2E: FF 36 0000 R                  push    nic_seg
04F1  2E: FF B7 0032 R                  push    rim_buf_offset[bx]      ;push (faraddr) bufptr
04F6  FF D0                             call    ax              ;call the C routine
04F8  83 C4 04                          add     sp,4

04FB  5A                                pop     dx              ;restore stuff
04FC  59                                pop     cx
04FD  07                                pop     es
04FE  1F                                pop     ds
04FF  C3                                ret

0500                    c_from_int      endp
```

```
                                page
                        ;-----------------------------------------------------------------------
                        ;
                        ;       Timer interrupt routine
                        ;
                        ; This is a periodic 55 msec interrupt called by the ROM BIOS.
                        ; We prescale by 4 to get a 220 msec periodic interrupt.
                        ; Then we call the l2_timerint() and l4_timerint() routines.
                        ;
                        ; All registers are saved, even though ax, dx, and ds have already
                        ; been saved by the first-level interrupt handler in the rom bios.
                        ; We do NOT switch stacks, so the running stack must be large enough
                        ; to accomodate whatever the timer interrupt routines use.
                        ;
                        ;-----------------------------------------------------------------------

                                public  timer_interrupt ;(for debugging only)
0500                    timer_interrupt proc    near

0500  2E: FE 0E 0017 R         dec     timer_prescale          ;count prescaler
                               $if     z                       ;prescaler counted down...

0507  2E: C6 06 0017 R 04              mov     timer_prescale,prescale ;reset prescaler

050D  50                               push    ax              ;save (almost) everything
050E  53                               push    bx              ;(C preserves si and di)
050F  51                               push    cx
0510  52                               push    dx
0511  55                               push    bp
0512  1E                               push    ds
0513  06                               push    es

0514  2E: 8E 1E 001C R                 mov     ds,c_dseg       ;setup ds = es = C's data segment
0519  2E: 8E 06 001C R                 mov     es,c_dseg

051E  E8 052C R                        call    l2_timerint     ;call Level 2 interrupt rtn
0521  E8 0000 E                        call    _l4_timerint    ;call Level 4 interrupt rtn

0524  07                               pop     es              ;restore everything
0525  1F                               pop     ds
0526  5D                               pop     bp
0527  5A                               pop     dx
0528  59                               pop     cx
0529  5B                               pop     bx
052A  58                               pop     ax
                               $endif
052B  CF                       iret

052C                    timer_interrupt endp
```

```
                                        page

                                ;-----------------------------------------------------------------------
                                ;
                                ;       l2_timerint();     Level 2 timer interrupt routine
                                ;
                                ; Check to see if a transmit command has taken too long.
                                ; If so, abort the transmit.  That will cause TA to come on
                                ; after the token is next received by this station, and then
                                ; a NIC interrupt will occur so that the next transmission can
                                ; be scheduled.
                                ;
                                ; Changes no registers.
                                ;
                                ;-----------------------------------------------------------------------

                                        public  l2_timerint     ;(for debugging only)
052C                            l2_timerint     proc    near

052C  2E: 80 3E 0018 R 00                       cmp     xmit_timer,0    ;are we transmitting?
                                                $ifnot  e               ;yes
0534  2E: FE 0E 0018 R                            dec   xmit_timer      ;count down
                                                    $if z               ;timeout!
053B  1E                                                push    ds
053C  2E: 8E 1E 0000 R                                  mov     ds,nic_seg              ;address nic
                                                        assume  ds:nic_map
0541  C6 06 0801 R 01                                   mov     rim_cmd,rimcmd_xmit_dis ;disable transmitter
0546  2E: 8E 1E 001C R                                  mov     ds,c_dseg               ;address C's data
                                                        assume  ds:dgroup
054B  FF 06 0000 E                                      inc     ds:_l4cnt_xmittimeout   ;count transmit timeout
054F  1F                                                pop     ds
                                                        assume  ds:nothing
                                                        ; we could call l4_trace to make a trace entry here,
                                                    $endif
                                                $endif
0550  C3                                        ret
0551                            l2_timerint     endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       Long-pointer memory move routine for C.
                        ;
                        ;
                        ; movel (from, to, length)
                        ;
                        ;   faraddr from, to;
                        ;   short int length;
                        ;
                        ;
                        ;  After our prologue, the stack looks like this:
                        ;
                        ;       12  length
                        ;       10  seg:to
                        ;        8  off:to
                        ;        6  seg:from
                        ;        4  off:from
                        ;        2  return address
                        ; bp--> 0  old bp
                        ;
                        ;-----------------------------------------------------------------------

                                public  _movel
0551                    _movel          proc    near
0551  55                                push    bp
0552  8B EC                            mov     bp,sp
0554  57                                push    di
0555  56                                push    si
0556  06                                push    es
0557  1E                                push    ds

0558  8B 76 04                         mov     si,[bp+4]       ;setup regs for movs
055B  8E 5E 06                         mov     ds,[bp+6]
055E  8B 7E 08                         mov     di,[bp+8]
0561  8E 46 0A                         mov     es,[bp+10]
0564  8B 4E 0C                         mov     cx,[bp+12]

0567  D1 E9                            shr     cx,1            ;# of words
0569  F3/ A5                           rep movsw               ;do words (faster than bytes)
                                        $if     c
056D  A4                                  movsb                 ;do extra odd byte
                                        $endif

056E  1F                                pop     ds
056F  07                                pop     es
0570  5E                                pop     si
0571  5F                                pop     di
0572  5D                                pop     bp
0573  C3                                ret
0574                    _movel          endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       XNS header word reversal routine
                        ;
                        ;
                        ; l2_reverse_xns (xptr)
                        ;
                        ;       farphaddr xptr;   /* pointer to XNS header in packet buffer */
                        ;
                        ;
                        ;  Reverse the following words in the XNS header pointed to by xptr:
                        ;
                        ;        xptr->length   (offset 6)
                        ;        xptr->dstskt   (offset 20)
                        ;        xptr->srcskt   (offset 32)
                        ;        xptr->seqno    (offset 40)
                        ;        ptr->ackno     (offset 42)
                        ;        ptr->allno     (offset 44)
                        ;
                        ;  Note that since we can't read C structure declarations, we have carnal
                        ;  knowledge of the above offsets.  But since they are fixed by XNS, they
                        ;  are not going to change!
                        ;
                        ;  After our prologue, the stack looks like this:
                        ;
                        ;       6  seg xptr
                        ;       4  offset xptr
                        ;       2  return address
                        ; bp--> 0  old bp
                        ;
                        ;-----------------------------------------------------------------------

                        swap            macro   arg             ;macro to swap memory word bytes
                                        mov     ax,arg
                                        mov     arg,ah
                                        mov     arg+1,al
                                        endm
                                        .xall   ;(see macro expansions)

                                        public  _l2_reverse_xns
0574                    _l2_reverse_xns proc    near
0574  55                                push    bp
0575  8B EC                            mov     bp,sp
0577  1E                                push    ds
0578  C5 5E 04                         lds     bx,[bp+4]

                                        swap    [bx]+6          ;reverse xptr->length
057B  8B 47 06                +         mov     ax,[bx]+6
057E  88 67 06                +         mov     [bx]+6,ah
0581  88 47 07                +         mov     [bx]+6+1,al
                                        swap    [bx]+20         ;reverse xptr->dskskt
0584  8B 47 14                +         mov     ax,[bx]+20
```

```
0587  88 67 14                +                   mov     [bx]+20,ah
058A  88 47 15                +                   mov     [bx]+20+1,al
                                                  swap    [bx]+32           ;reverse xptr->srcskt
058D  8B 47 20                +                   mov     ax,[bx]+32
0590  88 67 20                +                   mov     [bx]+32,ah
0593  88 47 21                +                   mov     [bx]+32+1,al
                                                  swap    [bx]+40           ;reverse xptr->seqno
0596  8B 47 28                +                   mov     ax,[bx]+40
0599  88 67 28                +                   mov     [bx]+40,ah
059C  88 47 29                +                   mov     [bx]+40+1,al
                                                  swap    [bx]+42           ;reverse xptr->ackno
059F  8B 47 2A                +                   mov     ax,[bx]+42
05A2  88 67 2A                +                   mov     [bx]+42,ah
05A5  88 47 2B                +                   mov     [bx]+42+1,al
                                                  swap    [bx]+44           ;reverse xptr->allno
05A8  8B 47 2C                +                   mov     ax,[bx]+44
05AB  88 67 2C                +                   mov     [bx]+44,ah
05AE  88 47 2D                +                   mov     [bx]+44+1,al

05B1  1F                                          pop     ds
05B2  5D                                          pop     bp
05B3  C3                                          ret
05B4                                _l2_reverse_xns endp
05B4                                _text         ends
                                                  end
```

Macros:

Name	Length
$DO	000E
$DOJCXZ	0001
$DOJMP	0001
$DOLOOP	0001
$DOUNTIL	0002
$DOWHILE	0002
$ELSE	0006
$ELSEIF	0008
$ELSEIFNOT	0008
$ENDIF	0009
$EXITIF	0004
$GETN	0001
$GETT	0001
$IF	0006
$IFNOT	0006
$JMP	0001
$LAB	0001
$PUTN	0001
$PUTT	0001
$REPEAT	0007
$REPEATLOOP	0007
$REPEATUNTIL	0007
$REPEATWHILE	0007
CLICK	0005
SWAP	0002

Segments and Groups:

Name	Size	Align	Combine	Class
DGROUP	GROUP			
_DATA	01B9	WORD	PUBLIC	'DATA'
NIC_MAP	1002	AT	D200	
_TEXT	05B4	BYTE	PUBLIC	'CODE'

Symbols:

Name	Type	Value	Attr	
BROADCAST_OK	Number	0000		
BUF0	Number	0000		
BUF1	Number	0002		
BUF2	Number	0004		
BUF3	Number	0006		
BUFF_STATE	L BYTE	0022	_TEXT	
CHECK_INTERRUPTS	L NEAR	038D	_TEXT	
C_DSEG	L WORD	001C	_TEXT	
C_FROM_INT	N PROC	04DE	_TEXT	Length =0022
DISKIO	Number	0013		
DOS_INT	Number	0021		

ENABLE_CMDS.	L BYTE	002A	_TEXT
END_TA	L NEAR	04AB	_TEXT
EOI.	Number	0020	
ERROR.	L NEAR	013C	_TEXT
GETVECTOR.	Number	0035	
IF$1000.	L NEAR	03DC	_TEXT
IF$1002.	L NEAR	03CF	_TEXT
IF$102	L NEAR	007F	_TEXT
IF$1052.	L NEAR	03F4	_TEXT
IF$1102.	L NEAR	0403	_TEXT
IF$1152.	L NEAR	041C	_TEXT
IF$1202.	L NEAR	044A	_TEXT
IF$1252.	L NEAR	044A	_TEXT
IF$1300.	L NEAR	0486	_TEXT
IF$1302.	L NEAR	0479	_TEXT
IF$1352.	L NEAR	04AB	_TEXT
IF$1402.	L NEAR	04AB	_TEXT
IF$1452.	L NEAR	04BC	_TEXT
IF$1502.	L NEAR	04CB	_TEXT
IF$152	L NEAR	01B5	_TEXT
IF$1552.	L NEAR	052B	_TEXT
IF$1602.	L NEAR	0550	_TEXT
IF$1652.	L NEAR	0550	_TEXT
IF$1702.	L NEAR	056E	_TEXT
IF$200	L NEAR	01B5	_TEXT
IF$202	L NEAR	01B0	_TEXT
IF$250	L NEAR	021F	_TEXT
IF$252	L NEAR	01FD	_TEXT
IF$300	L NEAR	021F	_TEXT
IF$302	L NEAR	0216	_TEXT
IF$350	L NEAR	0246	_TEXT
IF$352	L NEAR	0234	_TEXT
IF$400	L NEAR	0246	_TEXT
IF$402	L NEAR	023F	_TEXT
IF$452	L NEAR	0255	_TEXT
IF$500	L NEAR	02C7	_TEXT
IF$502	L NEAR	0270	_TEXT
IF$52.	L NEAR	0075	_TEXT
IF$552	L NEAR	02C7	_TEXT
IF$602	L NEAR	02C7	_TEXT
IF$650	L NEAR	02EE	_TEXT
IF$652	L NEAR	02DC	_TEXT
IF$700	L NEAR	02EE	_TEXT
IF$702	L NEAR	02E7	_TEXT
IF$752	L NEAR	02FD	_TEXT
IF$802	L NEAR	0362	_TEXT
IF$852	L NEAR	034C	_TEXT
IF$902	L NEAR	03F4	_TEXT
IF$952	L NEAR	03AC	_TEXT
IF$L	Number	0000	
IF$N	Number	06A4	
IF$NS.	Number	06A6	
IF$NS1	Number	06A4	
IF$NS2	Number	0672	

```
IF$NS3 . . . . . .          Number  0258
IF$T . . . . . .            Number  0002
IF$T1. . . . . .            Number  0002
IF$T2. . . . . .            Number  0002
IF$T3. . . . . .            Number  0002
INIT_EXIT. . . . . .        L NEAR  0144    _TEXT
INIT_OK. . . . . .          L NEAR  0141    _TEXT
INTMASKSV. . . . . .        L BYTE  0015    _TEXT
INT_SP . . . . . .          L WORD  0020    _TEXT
INT_SS . . . . . .          L WORD  001E    _TEXT
INT_STACK. . . . . .        E NEAR  012B    _DATA
INT_STACK_BEG. . . . . .    L BYTE  0000    _DATA   Length =012C
IRQCMD . . . . . .          Number  0020
IRQMASK. . . . . .          Number  0021
L2_DATA. . . . . .          N PROC  0000    _TEXT   Global  Length =003A
L2_TIMERINT. . . . . .      N PROC  052C    _TEXT   Global  Length =0025
L4_IN_USE. . . . . .        Number  0008
L4_LONG_MODE . . . . . .    Number  0009
L4_OUR_ETHER . . . . . .    Number  000A
L4_PUBLICS . . . . . .      L DWORD 0008    _TEXT
NICINTSV . . . . . .        L DWORD 000D    _TEXT
NIC_CTL. . . . . .          L BYTE  0802    NIC_MAP
NIC_INT. . . . . .          L BYTE  0803    NIC_MAP
NIC_INTERRUPT. . . . . .    N PROC  0365    _TEXT   Global  Length =0179
NIC_INT_LEVEL. . . . . .    L BYTE  000C    _TEXT
NIC_LOCATION . . . . . .    Number  D200
NIC_ROM. . . . . .          L WORD  1000    NIC_MAP
NIC_SEG. . . . . .          L WORD  0000    _TEXT
NIC_SIGNATURE. . . . . .    Number  AA55
NIC_STAT . . . . . .        Alias   NIC_CTL
N_OWED_BUFFS . . . . . .    L BYTE  001A    _TEXT
N_XMIT_RETRIES . . . . . .  L BYTE  0019    _TEXT
OTHER_BUFFER . . . . . .    Number  0002
PAGE0. . . . . .            L BYTE  0000    NIC_MAP Length =0200
PAGE1. . . . . .            L BYTE  0200    NIC_MAP Length =0200
PAGE2. . . . . .            L BYTE  0400    NIC_MAP Length =0200
PAGE3. . . . . .            L BYTE  0600    NIC_MAP Length =0200
PANIC1 . . . . . .          L BYTE  012C    _DATA
PANIC2 . . . . . .          L BYTE  0145    _DATA
PANIC3 . . . . . .          L BYTE  015F    _DATA
PANIC4 . . . . . .          L BYTE  017B    _DATA
PANIC5 . . . . . .          L BYTE  019A    _DATA
POR. . . . . .              Number  0010
PRESCALE . . . . . .        Number  0004
RCV_EMPTY. . . . . .        Number  0005
RCV_ENABLED. . . . . .      Number  0006
RCV_FULL . . . . . .        Number  0007
RCV_INL4 . . . . . .        Number  0008
RECON. . . . . .            Number  0004
RI . . . . . .              Number  0080
RIMCMD_CLRFLGS . . . . . .  Number  001E
RIMCMD_CONFIG_L. . . . . .  Number  000D
RIMCMD_CONFIG_S. . . . . .  Number  0005
RIMCMD_RECV_EN . . . . . .  Number  0004
```

```
RIMCMD_XMIT_DIS. . . . . .      Number  0001
RIMCMD_XMIT_EN . . . . . .      Number  0003
RIM_BUF_OFFSET . . . . . .      L WORD  0032    _TEXT
RIM_CMD. . . . . .              L BYTE  0801    NIC_MAP
RIM_INT. . . . . .              L BYTE  0800    NIC_MAP
RIM_INTERRUPT. . . . . .        Number  0008
RIM_INT_COPY . . . . . .        L BYTE  001B    _TEXT
RIM_INT_ENB. . . . . .          Number  0004
RIM_STAT . . . . . .            Alias   RIM_INT
SETVECTOR. . . . . .            Number  0025
SPKRSV . . . . . .              L BYTE  0016    _TEXT
SPKR_ENABLE. . . . . .          Number  0002
SPKR_PORT. . . . . .            Number  0061
TA . . . . . .                  Number  0001
TIMER_GATE . . . . . .          Number  0001
TIMER_INT. . . . . .            Number  001C
TIMER_INTERRUPT. . . . . .      N PROC  0500    _TEXT   Global  Length =002C
TIMER_PRESCALE . . . . . .      L BYTE  0017    _TEXT
TIMINTSV . . . . . .            L DWORD 0011    _TEXT
TMA. . . . . .                  Number  0002
TO_XMIT_PKT. . . . . .          Number  000F
XMIT_EMPTY . . . . . .          Number  0001
XMIT_ENABLED . . . . . .        Number  0004
XMIT_FULL. . . . . .            Number  0003
XMIT_INL4. . . . . .            Number  0002
XMIT_RETRIES . . . . . .        Number  0001
XMIT_TIMER . . . . . .          L BYTE  0018    _TEXT
XNSADDR. . . . . .              L WORD  0002    _TEXT
_L2_BUFF_STATE . . . . . .      E BYTE  0022    _TEXT   Global
_L2_GETBUF . . . . . .          N PROC  01DF    _TEXT   Global  Length =0043
_L2_INIT . . . . . .            N PROC  003A    _TEXT   Global  Length =012A
_L2_RCVRELEASE . . . . . .      N PROC  02CB    _TEXT   Global  Length =009A
_L2_REVERSE_XNS. . . . . .      N PROC  0574    _TEXT   Global  Length =0040
_L2_SENDBUF. . . . . .          N PROC  0222    _TEXT   Global  Length =00A9
_L2_TERMINATE. . . . . .        N PROC  0164    _TEXT   Global  Length =007B
_L4CNT_L2INT_RECON . . . . . .  V WORD  0000    _DATA   External
_L4CNT_L2INT_RI. . . . . .      V WORD  0000    _DATA   External
_L4CNT_L2INT_TA. . . . . .      V WORD  0000    _DATA   External
_L4CNT_P_RETRIES . . . . . .    V WORD  0000    _DATA   External
_L4CNT_XMITNOACK . . . . . .    V WORD  0000    _DATA   External
_L4CNT_XMITTIMEOUT . . . . . .  V WORD  0000    _DATA   External
_L4_GOTBUF . . . . . .          L NEAR  0000    _TEXT   External
_L4_RCVINTR. . . . . .          L NEAR  0000    _TEXT   External
_L4_TIMERINT . . . . . .        L NEAR  0000    _TEXT   External
_MOVEL . . . . . .              N PROC  0551    _TEXT   Global  Length =0023
_PANIC . . . . . .              L NEAR  0000    _TEXT   External

39358 Bytes free

Warning Severe
Errors  Errors
0       0
```