```
                page    58,132
;-----------------------------------------------------------------------------
;
;file: ql4.asm
;
;
;    This is a temporary interface between the interrupt-driven ql4_ calls
;    made by the Queen and the PC L4 calls as of SR4, 3/14/85.
;
;    The entry comments on the procedures, though, are valid documentation
;    of the assumptions made by the Queen.
;
;    External routine names have been truncation to 8 characters because
;    that's what the Lattice compiler does.
;
;       ALL CODE AND STRUCTURES HERE REFLECT THE SMALL C MODEL!!!
;       DATA AND CODE SEGMENTS ARE DIFFERENT!!!
;
;-----------------------------------------------------------------------------
;
;       How it works
;
; We are passed the address of a "Queen Request Block" (qrb) which contains
; parameters for the transport level operations.  In order to establish a
; connection, we get a transport level RB ("real L4 rb") and point the qrb
; to it.  The qrbs which we are responsible for are linked together with
; the "qrb.l4link" field, unused by the Queen.
;
; When the "wait for interrupt" routine is called by the queen, we make
; repeated calls to the L4 churn routine and scan our chain of qrbs looking
; for a L4 rb which has completed its operation.  (We maintain an internal
; state flag in the qrb to indicate which operation is in progress for that
; rb.)  When the operation is complete, we post the mailbox whose address
; is in the qrb with the address of the qrb and return to the dispatcher.
;
; We also return to the dispatcher if the ready queue has been added to by
; any other interrupt routine, such as the timer.
;
; When the connection is broken by an error, a call to l4_disconn, or a
; call to l4_abort, the L4 rb is freed.
;
;-----------------------------------------------------------------------------
;
;       Change log
;
;  4/xx/85  L. Shustek  Initial versions.
;
;  7/03/85  L. Shustek  Add ql4_conn to initiate connections or send
;                       broadcast messages. (For pc network netbios/smb.)
;
;  7/11/85  L. Shustek  Allow broadcast reception.
;                       Add speaker click for network activity.
;                       Add support for two well-known sockets for jdw's smbs.
```

```
                        ;
                        ;-----------------------------------------------------------------------------

                                .sall                   ;supress macro expansions

                        ;;      include o:sm8086.mac    ;Lattice C small model macros
                                .list

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

= 0000                  on_nic          equ     0
= 0000                  l4_in_our_seg   equ     0

                        ;;      include e:l4asm.itf             ;level 4 interface
                                .list



                        ;
                        ;    The simplified "Queen L4 Request Block" - qrb
                        ;
                        ;       This must match the C declaration of the same structure.
                        ;

                        qrb             struc
0000  ????              qrb_id          dw      ?       ;'RB'
0002  ????              qrb_mlink       dw      ?       ;link field
0004  ????              qrb_mail        dw      ?       ;ptr to mailbox to post
0006  ????????          qrb_rb          dd      ?       ;long ptr to real L4 rb  (private to us)
000A  ?? ??             qrb_state       db      ?,?     ;internal state: st_xxx  (private to us)
000C  ????              qrb_l4link      dw      ?       ;for us to link qrbs     (private to us)
000E  ????              qrb_status      dw      ?       ;ending status: l4st_xxx
0010  ????              qrb_churncnt    dw      ?       ;churn counter (for debugging)
0012  ????              qrb_rcvptr      dw      ?
0014  ????              qrb_rcvlength   dw      ?
0016  ????              qrb_rcvlimit    dw      ?
0018  ????              qrb_sndptr      dw      ?
001A  ????              qrb_sndlength   dw      ?
001C  ??                qrb_sndtype     db      ?
001D  ??                qrb_rcvtype     db      ?
001E  ????              qrb_wks         dw      ?       ;well-known socket
0020  ??                qrb_pkthdr      db      ?       ; struc ether_header starts here
0021                    qrb             ends



                        ;
                        ;       The qrb_status return values
                        ;

= 0000                  l4st_uncon      equ     0       ;no connection established (anymore)
= 0001                  l4st_busy       equ     1       ;command still in progress; still connected
= 0002                  l4st_done       equ     2       ;command terminated ok; still connected
```

```
                        level_four_interface

= 0003                  l4st_partial    equ     3       ;command needs more buffer (NOT IMPLEMENTED)
= 0004                  l4st_failed     equ     4       ;command failed; still connected



                        ;
                        ;       Our internal qrb_state values  (must be even for jump table index)
                        ;

= 0000                  st_idle         equ     0       ;connected but idle
= 0002                  st_openrcv      equ     2       ;openreceiving (awaiting connection)
= 0004                  st_rcv          equ     4       ;receiving
= 0006                  st_snd          equ     6       ;sending or connecting
= 0008                  st_discon       equ     8       ;disconnected
= 000A                  st_sendack      equ     10      ;sending an ack after openreceive



                        ;
                        ;               DS-based variables
                        ;

                                        dseg
                                        extrn   ready_tc:word           ;ready_tcb: head of ready list
0000  0000              qrb_list        dw      0                       ;head of qrb chain
                                        public  qrb_list                ; (for debug monitor)


                                        endds


                        ;
                        ;       Miscellanea
                        ;

= 000D                  cr              equ     13
= 000A                  lf              equ     10
= 001B                  esc             equ     27

= 0021                  dos_int         equ     21h
= 0009                  dosint_prints   equ     09h                     ;print string at ds:dx until '$'

= 0016                  keyboard        equ     16h                     ;keyboard read/status (ah=0/1)
= 0010                  video           equ     10h                     ;screen write tty (ah=14)

= 0061                  spkr_port       equ     61h                     ;speaker I/O port


                                        pseg                            ;start the code segment

                        ;;              include e:l4loc.asm
                                        .list

                                        assume  ds:dgroup               ;(cancelled by l4loc.asm)
```

```
                                level_four_interface

                                        extrn   exit:near


                                ;
                                ;               CS-based variables
                                ;

00B5  00                        interrupt       db      0                       ;did we see an "interrupt"?
00B6  01                        key_count       db      1                       ;keyboard check countdown
00B7  0000                      ourwks1         dw      0                       ;our two well-known sockets
00B9  0000                      ourwks2         dw      0

00BB  51 4C 34 3A 20 4C         msg_nwinit      db      'QL4: L4_locate error',cr,lf,'$'
      34 5F 6C 6F 63 61
      74 65 20 65 72 72
      6F 72 0D 0A 24
00D2  51 4C 34 3A 20 4E         msg_rb          db      'QL4: No rbs available.',cr,lf,'$'
      6F 20 72 62 73 20
      61 76 61 69 6C 61
      62 6C 65 2E 0D 0A
      24
00EB  51 4C 34 3A 20 42         msg_socket      db      'QL4: Bad l4_listen',cr,lf,'$'
      61 64 20 6C 34 5F
      6C 69 73 74 65 6E
      0D 0A 24
0100  51 4C 34 3A 20 73         msg_notidle     db      'QL4: send/rcv/disconn when not idle',cr,lf,'$'
      65 6E 64 2F 72 63
      76 2F 64 69 73 63
      6F 6E 6E 20 77 68
      65 6E 20 6E 6F 74
      20 69 64 6C 65 0D
      0A 24
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_init();
                        ;
                        ; Initialize the transport level.
                        ; Returns immediately and doesn't post the mailbox.
                        ;

                                public  ql4_init
0126                    ql4_init        proc    near
0126  1E                               push    ds
0127  06                               push    es
                                        l4_call l4locate                ;find transport-level routines
012B  07                               pop     es
012C  1F                               pop     ds
012D  3C 00                            cmp     al,l4_ok
                                        $ifnot  e
0131  E9 04EA R                           jmp   error_nwinit
                                        $endif
0134  C3                               ret
0135                    ql4_init        endp



                        ;
                        ;       ql4_listen (wks)
                        ;
                        ;  Allow incoming connections on the specified well-known socket.
                        ;  We support two sockets.
                        ;  Returns immediately and doesn't post the mailbox.
                        ;

                                public  ql4_list;en
0135                    ql4_list        proc    near
0135  55                               push    bp
0136  8B EC                           mov     bp,sp
0138  2E: A1 00B7 R                    mov     ax,ourwks1              ;save last 2 wks's listened on
013C  2E: A3 00B9 R                    mov     ourwks2,ax
0140  8B 46 04                         mov     ax,[bp+4]               ;get argument: wks
0143  2E: A3 00B7 R                    mov     ourwks1,ax              ;save it for later
                                        l4_call ignore                  ; (clear any old listens first)
014C  2E: A1 00B7 R                    mov     ax,ourwks1
0150  B3 01                            mov     bl,1                    ;broadcast is ok
                                        l4_call listen
0157  3C 00                            cmp     al,sock_ok
                                        $ifnot  e
015B  E9 04F6 R                           jmp   error_socket
                                        $endif
015E  5D                               pop     bp
015F  C3                               ret
0160                    ql4_list        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_abort (&qrb)
                        ;
                        ;  Abort the current connection.
                        ;  Returns immediately and doesn't post the mailbox.
                        ;

                                public  ql4_abor;t
0160                    ql4_abor        proc    near
0160  55                               push    bp
0161  8B EC                           mov     bp,sp
0163  8B 7E 04                        mov     di,[bp+4]               ;get &qrb
0166  C4 75 06                        les     si,[di].qrb_rb          ;es:si is the real l4 rb
0169  E8 0476 R                       call    abort                   ;abort, and free_rb
016C  5D                               pop     bp
016D  C3                               ret
016E                    ql4_abor        endp
```

! save ds!

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_openrcv ( &qrb )
                        ;
                        ; Setup to accept an incoming connection for the specified qrb.
                        ; After getting a L4 rb and linking the qrb, this returns immediately.
                        ; When a connection is later discovered to be incoming, the qrb is mailed
                        ; to the mailbox whose address is contained therein.
                        ;
                        ;  qrb fields set after post:   wks
                        ;                               pkthdr.dest_host
                        ;

                                public  ql4_open;rcv
016E                    ql4_open        proc    near
016E  55                                push    bp
016F  8B EC                             mov     bp,sp
0171  06                                push    es

0172  1E                                push    ds
                                          l4_call activate_rb           ;get an l4 rb
0178  8C D8                               mov   ax,ds
017A  8E C0                               mov   es,ax                   ;l4 rb address in es:si
017C  1F                                pop     ds
017D  0B C0                             or      ax,ax                   ;got one?
                                          $if   z
0181  07                                    pop es
0182  E9 04F0 R                             jmp error_rb                ;no: fatal error
                                          $endif

0185  8B 7E 04                          mov     di,[bp+4]               ;get &qbb
0188  C7 45 10 0000                     mov     word ptr [di].qrb_churncnt,0  ; zero churn counter
018D  89 75 06                          mov     word ptr [di].qrb_rb,si ;point our qrb to the real rb
0190  8C 45 08                          mov     word ptr [di].qrb_rb+2,es

0193  C6 45 0A 02                       mov     [di].qrb_state,st_openrcv  ;state is "do open_rcv"
0197  C7 45 0E 0001                     mov     [di].qrb_status,l4st_busy  ;status is "busy"

019C  A1 0000 R                         mov     ax,qrb_list             ;link us onto the chain of qrbs
019F  89 45 0C                          mov     [di].qrb_l4link,ax
01A2  89 3E 0000 R                      mov     qrb_list,di

01A6  07                                pop     es
01A7  5D                                pop     bp                      ;return
01A8  C3                                ret
01A9                    ql4_open        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_conn (&qrb)
                        ;
                        ; Establish a connection and send an initial message
                        ;
                        ;       qrb.dest_host is the destination XNS address, or all ones for broadcast.
                        ;       qrb.sndptr    is the buffer address
                        ;       qrb.sndlength is the message size
                        ;       qrb.sndtype   is the message type
                        ;       qrb.wks       is the socket to send on
                        ;
                        ; When the message is sent, the qrb is mailed to the mailbox whose
                        ; address is in qrb.mail.
                        ;
                        ;       qrb.status      will indicate if it was successful or not.
                        ;                       Broadcast or failure will set it to l4_uncon
                        ;                       Successful non-broadcast will set it to l4_done.
                        ;

01A9                    ql4_conn        public  ql4_conn
01A9                                    proc    near
01A9  55                                push    bp
01AA  8B EC                             mov     bp,sp
01AC  06                                push    es

01AD  1E                                push    ds
                                          l4_call activate_rb           ;get an l4 rb
01B3  8C D8                               mov   ax,ds
01B5  8E C0                               mov   es,ax                   ;l4 rb address in es:si
01B7  1F                                pop     ds
01B8  0B C0                             or      ax,ax                   ;got one?
01BC  07                                  $if   z
01BD  E9 04F0 R                             pop es
                                            jmp error_rb                ;no: fatal error
                                          $endif

01C0  8B 7E 04                          mov     di,[bp+4]               ;get &qbb
01C3  C7 45 10 0000                     mov     word ptr [di].qrb_churncnt,0  ; zero churn counter
01C8  89 75 06                          mov     word ptr [di].qrb_rb,si ;point our qrb to the real rb
01CB  8C 45 08                          mov     word ptr [di].qrb_rb+2,es

01CE  A1 0000 R                         mov     ax,qrb_list             ;link us onto the chain of qrbs
01D1  89 45 0C                          mov     [di].qrb_l4link,ax
01D4  89 3E 0000 R                      mov     qrb_list,di

01D8  8D 5D 20                          lea     bx,[di].qrb_pkthdr          ;move XNS address to real rb
01DB  8B 47 0E                          mov     ax,[bx].dest_host
01DE  26: 89 44 1E                      mov     es:[si].hdr_dest_host,ax
01E2  8B 47 10                          mov     ax,[bx].dest_host+2
01E5  26: 89 44 20                      mov     es:[si].hdr_dest_host+2,ax
01E9  8B 47 12                          mov     ax,[bx].dest_host+4
01EC  26: 89 44 22                      mov     es:[si].hdr_dest_host+4,ax
```

```
01F0  8B 45 1E                          mov     ax,[di].qrb_wks         ;copy wks
01F3  26: 89 44 24                      mov     es:[si].hdr_dest_socket,ax
01F7  8B 45 18                          mov     ax,[di].qrb_sndptr      ;copy send ptr
01FA  26: 89 44 0A                      mov     word ptr es:[si].send_ptr,ax
01FE  26: 8C 5C 0C                      mov     word ptr es:[si].send_ptr+2,ds
0202  8B 45 1A                          mov     ax,[di].qrb_sndlength   ;copy buffer size
0205  26: 89 44 0E                      mov     es:[si].send_length,ax
0209  8A 45 1C                          mov     al,[di].qrb_sndtype     ;copy type
020C  26: 88 44 33                      mov     es:[si].hdr_data_type,al

0210  C6 45 0A 06                      mov     [di].qrb_state,st_snd   ;state is "sending"
0214  C7 45 0E 0001                     mov     [di].qrb_status,l4st_busy  ;status is "busy"

0219  1E                                push    ds
021A  8C C0                               mov   ax,es
021C  8E D8                               mov   ds,ax
                                          l4_call connect               ;start it, rb in ds:si
0223  E8 04DF R                           call  click
0226  1F                                pop     ds

0227  07                                pop     es
0228  5D                                pop     bp                     ;return
0229  C3                                ret
022A                            q14_conn        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_rcvmsg (&qrb)
                        ;
                        ; Start receiving a message.
                        ;
                        ; Input: qrb.rcvptr    is the buffer address
                        ;        qrb.rcvlimit  is the buffer size
                        ;
                        ; If a complete message is already received, the qrb.status will indicate
                        ; whether is was successful and values will be returned as shown below.
                        ;
                        ; If the message is not yet received, the qrb.status will be set to l4st_busy.
                        ; When the message is received, the qrb is mailed to the mailbox whose
                        ; address is in qrb.mail.
                        ;
                        ; Output:  qrb.status      will indicate if it was successful
                        ;          qrb.rcvlength   will be the actual message length
                        ;          qrb.type        will be the message type
                        ;
                        ; Note: This routine duplicates the essence of some code from state_rcv.
                        ;

                                        public  ql4_rcvm;sg
022A                    ql4_rcvm        proc    near
022A  55                                push    bp
022B  8B EC                             mov     bp,sp
022D  06                                push    es

022E  8B 7E 04                          mov     di,[bp+4]               ;get &qrb in ds:di
0231  80 7D 0A 00                       cmp     [di].qrb_state,st_idle  ;better be idle
                                        $ifnot  e
0237  E9 04FC R                           jmp   err_not_idle
                                        $endif
023A  C7 45 10 0000                     mov     word ptr [di].qrb_churncnt,0  ; zero churn counter
023F  C4 75 06                          les     si,[di].qrb_rb          ;get real rb in es:si
0242  8B 45 12                          mov     ax,[di].qrb_rcvptr      ;copy receive ptr
0245  26: 89 44 40                      mov     word ptr es:[si].recv_ptr,ax
0249  26: 8C 5C 42                      mov     word ptr es:[si].recv_ptr+2,ds
024D  8B 45 16                          mov     ax,[di].qrb_rcvlimit    ;copy buffer size
0250  26: 89 44 44                      mov     es:[si].recv_limit,ax
0254  C6 45 0A 04                       mov     [di].qrb_state,st_rcv   ;state is "receiving"
0258  C7 45 0E 0001                     mov     [di].qrb_status,l4st_busy  ;status is "busy"

025D  1E                                push    ds
025E  8C C0                               mov   ax,es
0260  8E D8                               mov   ds,ax
                                          l4_call recv_msg              ;start it, rb in ds:si
0267  E8 04DF R                           call  click
026A  1F                                pop     ds
                        ;
                        ; This is the check to see if the message is already received.
                        ; This enhances performance in the case of alternating sends and
```

```
                                level_four_interface

                                ; receives where the ACK for the send is piggybacked.
                                ;
026B  26: 8A 44 3E                              mov     al,es:[si].recv_status  ;get the receive status
026F  3C 05                                     cmp     al,tf_in_prog
                                                $ifnot  e
0273  3C 00                                       cmp   al,tf_idle
                                                  $if   e
0277  C7 45 0E 0002                                 mov [di].qrb_status,l4st_done   ;ended ok; set status
027C  C6 45 0A 00                                   mov [di].qrb_state,st_idle      ;state is idle
0280  26: 8B 44 46                                  mov ax,es:[si].recv_length      ;move length to our rb
0284  89 45 14                                      mov [di].qrb_rcvlength,ax
0287  26: 8A 44 3F                                  mov al,es:[si].recv_type        ;move type to our rb
028B  88 45 1D                                      mov [di].qrb_rcvtype,al
                                                  $else
0291  E8 0476 R                                     call abort          ;ended badly, abort
0294  C6 45 0A 00                                   mov [di].qrb_state,st_idle          ;state is idle
0298  E9 034D R                                     jmp  search_qrbs
                                                  $endif
                                                $endif

029B  07                                        pop     es                      ;return
029C  5D                                        pop     bp
029D  C3                                        ret
029E                            ql4_rcvm        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_sndmsg (&qrb)
                        ;
                        ; Start sending a message.
                        ;
                        ;       qrb.sndptr    is the buffer address
                        ;       qrb.sndlength is the message size
                        ;       qrb.sndtype   is the message type
                        ;
                        ; When the message is sent, the qrb is mailed to the mailbox whose
                        ; address is in qrb.mail.
                        ;
                        ;       qrb.status      will indicate if it was successful or not
                        ;

                                public  ql4_sndm;sg
029E                    ql4_sndm        proc    near
029E  55                                push    bp
029F  8B EC                             mov     bp,sp
02A1  06                                push    es

02A2  8B 7E 04                          mov     di,[bp+4]               ;get &qrb
02A5  80 7D 0A 00                       cmp     [di].qrb_state,st_idle  ;better be idle
                                        $ifnot  e
02AB  E9 04FC R                           jmp   err_not_idle
                                        $endif
02AE  C7 45 10 0000                     mov     word ptr [di].qrb_churncnt,0  ; zero churn counter
02B3  C4 75 06                          les     si,[di].qrb_rb          ;get real rb in es:si
02B6  8B 45 18                          mov     ax,[di].qrb_sndptr      ;copy send ptr
02B9  26: 89 44 0A                      mov     word ptr es:[si].send_ptr,ax
02BD  26: 8C 5C 0C                      mov     word ptr es:[si].send_ptr+2,ds
02C1  8B 45 1A                          mov     ax,[di].qrb_sndlength   ;copy buffer size
02C4  26: 89 44 0E                      mov     es:[si].send_length,ax
02C8  8A 45 1C                          mov     al,[di].qrb_sndtype     ;copy type
02CB  26: 88 44 33                      mov     es:[si].hdr_data_type,al
02CF  C6 45 0A 06                       mov     [di].qrb_state,st_snd   ;state is "sending"
02D3  C7 45 0E 0001                     mov     [di].qrb_status,l4st_busy  ;status is "busy"

02D8  1E                                push    ds
02D9  8C C0                               mov   ax,es
02DB  8E D8                               mov   ds,ax
                                          l4_call send_msg              ;start it, rb in ds:si
02E2  E8 04DF R                           call  click
02E5  1F                                pop     ds

02E6  07                                pop     es                      ;and return
02E7  5D                                pop     bp
02E8  C3                                ret
02E9                    ql4_sndm        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       ql4_disconn (&qrb)
                        ;
                        ; Start a disconnect.
                        ;
                        ; When the disconnect is complete, the qrb is mailed to the mailbox whose
                        ; address is in qrb.mail.
                        ;
                        ;       qrb.status      will indicate if it was successful or not
                        ;

                                public  ql4_disc;onn
02E9                    ql4_disc        proc    near
02E9  55                                push    bp
02EA  8B EC                             mov     bp,sp

02EC  8B 7E 04                          mov     di,[bp+4]               ;get &qrb
02EF  80 7D 0A 00                       cmp     [di].qrb_state,st_idle  ;better be idle
                                        $ifnot  e
02F5  E9 04FC R                           jmp   err_not_idle
                                        $endif
02F8  C7 45 10 0000                     mov     word ptr [di].qrb_churncnt,0  ; zero churn counter
02FD  C6 45 0A 08                       mov     [di].qrb_state,st_discon    ;state is "disconnecting"
0301  C7 45 0E 0001                     mov     [di].qrb_status,l4st_busy   ;status is "busy"
0306  1E                                push    ds
0307  C5 75 06                           lds     si,[di].qrb_rb          ;get real rb in ds:si
                                        l4_call disconn                 ;start the disconnect
030F  E8 04DF R                         call    click
0312  1F                                pop     ds

0313  5D                                pop     bp
0314  C3                                ret
0315                    ql4_disc        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       wait_intr()
                        ;
                        ; Wait for an interrupt.
                        ;
                        ; We really just sit churning L4 until one of our rbs is done,
                        ; then post the mailbox which belongs to it and return to the dispatcher.
                        ;
                        ; Each qrb contains an internal state variable which indicates what it is
                        ; waiting for.  The state action routine is entered with registers as follows:
                        ;
                        ;       ds:di  points to the qrb
                        ;       es:si  points to the real L4 rb
                        ;
                        ; The state action routines normally return to "state_done".  If, however,
                        ; the qrb is taken off the list, then it return to "search_qrbs" to start
                        ; back at the beginning of the list.
                        ;
                        ;
                        ; The state variable is an index into the following table
                        ; of state action routines:
                        ;

0315                    jump_table      label   word
0315                                    org     jump_table+st_idle
0315  0361 R                           dw      state_done
0317                                    org     jump_table+st_openrcv
0317  0379 R                           dw      state_openrcv
0319                                    org     jump_table+st_rcv
0319  03F4 R                           dw      state_rcv
031B                                    org     jump_table+st_snd
031B  0425 R                           dw      state_snd
031D                                    org     jump_table+st_discon
031D  045D R                           dw      state_disconn
031F                                    org     jump_table+st_sendack
031F  03D1 R                           dw      state_sendack


                                        public  wait_int;r
0321                    wait_int        proc    near

0321                    churn:          l4_call l4churn         ;Get l4 to do something

= 0023                  tr_l4churn      equ     35              ;TEMP: trace(tr_l4churn,(lword)junk);
                        ;;;             push    ds              ;junk
                        ;;;             push    cs              ;junk
                        ;;;             mov     ax,tr_l4churn
                        ;;;             push    ax
                        ;;;             extrn   do_trace:near
                        ;;;             call    do_trace
                        ;;;             add     sp,6
```

```
                        level_four_interface

0326  2E: FE 0E 00B6 R                dec     key_count               ;time to check keyboard?
                                      $if     z
032D  2E: C6 06 00B6 R 64                     mov     key_count,100   ;yes:reset counter
0333  B4 01                                   mov     ah,1            ;check for keyboard key
0335  CD 16                                   int     keyboard
                                              $ifnot  z
0339  B4 00                                     mov   ah,0            ;got one: read to purge it
033B  CD 16                                     int   keyboard
033D  50                                        push  ax
                                                ;;mov ah,14           ;DON'T echo to screen
                                                ;;int video
033E  58                                        pop   ax
033F  3C 1B                                     cmp   al,esc          ;check for ESC
                                                $if   e
                                                 extrn debug:near
0343  E8 0000 E                                  call  debug          ;enter the debug monitor
                                                $endif
                                              $endif
                                      $endif

0346  06                                      push    es              ;---------------------- save es
0347  2E: C6 06 00B5 R 00                     mov     interrupt,0     ;flag "no interrupt yet"

034D                                  search_qrbs:
034D  8B 3E 0000 R                            mov     di,qrb_list     ;head of our qrb chain

                                              $do                     ;for all qrbs on the chain
0351  FF 45 10                                  inc   word ptr [di].qrb_churncnt  ;bump churn counter
0354  C4 75 06                                  les   si,[di].qrb_rb          ;es:si is the real 14 rb
0357  8A 5D 0A                                  mov   bl,[di].qrb_state       ;index into jump table by state
035A  B7 00                                     mov   bh,0
035C  2E: FF A7 0315 R                          jmp   jump_table[bx]          ;go to state routine
0361                                  state_done:
0361  8B 7D 0C                                 mov   di,[di].qrb_14link      ;next qrb
0364  0B FF                                     or    di,di
                                              $repeatuntil z          ;until end of the qrb chain
0368  07                                      pop     es              ;---------------------- restore es

0369  83 3E 0000 E 00                   cmp     ready_tc,0      ;did an interrupt rtn make a task ready?
                                        $if     z               ;if not,
0370  2E: 80 3E 00B5 R 00                 cmp   interrupt,0     ;continue until something happens here
0376  74 A9                               je    churn
                                        $endif
0378  C3                                ret                     ;otherwise return to the queen
```

```
                        level_four_interface

                                page

                        ;
                        ;       openrcv state
                        ;

0379  1E                state_openrcv:  push    ds              ;dseg unavailable!!!
037A  8C C0                             mov     ax,es
037C  8E D8                             mov     ds,ax           ;14 rb in ds:si

037E  2E: A1 00B7 R                     mov     ax,ourwks1      ;check first socket
0382  26: 89 44 30                      mov     es:[si].hdr_src_socket,ax  ;put into real rb
                                        14_call open_recv       ;check for incoming connection
0388  72 16                             jc      got_incoming

038D  2E: A1 00B9 R                     mov     ax,ourwks2      ;check second socket, if any
0391  0B C0                             or      ax,ax
                                        $ifnot  z
0395  26: 89 44 30                        mov   es:[si].hdr_src_socket,ax  ;put into real rb
                                          14_call open_recv     ;check for incoming connection
039E  72 03                               jc    got_incoming
                                        $endif

03A0  1F                                pop     ds
03A1  EB BE                             jmp     state_done      ;nothing either way


03A3                    got_incoming:
                                          14_call ack_now       ;we got one: start the ack going
03A8  E8 04DF R                           call  click
03AB  1F                                  pop   ds
03AC  C6 45 0A 0A                         mov   [di].qrb_state,st_sendack  ;state is "sending ack"
03B0  26: 8B 44 30                        mov   ax,es:[si].hdr_src_socket  ;move wks to qrb
03B4  89 45 1E                            mov   [di].qrb_wks,ax
03B7  8D 5D 20                            lea   bx,[di].qrb_pkthdr         ;move XNS address to qrb
03BA  26: 8B 44 1E                        mov   ax,es:[si].hdr_dest_host
03BE  89 47 0E                            mov   [bx].dest_host,ax
03C1  26: 8B 44 20                        mov   ax,es:[si].hdr_dest_host+2
03C5  89 47 10                            mov   [bx].dest_host+2,ax
03C8  26: 8B 44 22                        mov   ax,es:[si].hdr_dest_host+4
03CC  89 47 12                            mov   [bx].dest_host+4,ax
03CF  EB 90                               jmp   state_done      ;exit without posting to await ack send
                        ; When we are truly interrupt-driven, we can post directly from here so that
                        ; processing the request starts before the ack is sent, since the interrupt
                        ; routines will see to it that the ack goes out even if the Queen is kept busy.



                        ;
                        ;       sendack state
                        ;

03D1  26: 8A 44 3E      state_sendack:  mov     al,es:[si].recv_status
```

level_four_interface

```
03D5  3C 05                                   cmp     al,tf_in_prog
                                              $ifnot  e                       ;done
03D9  3C 00                                     cmp   al,tf_idle
                                                $if   e
03DD  C7 45 0E 0002                               mov [di].qrb_status,l4st_done       ;ended ok, set status
03E2  E8 04C3 R                                   call post                           ;post the task
                                                $else
03E8  E8 0476 R                                   call abort                          ;ended badly, abort
03EB  E8 04C3 R                                   call post
03EE  E9 034D R                                   jmp  search_qrbs
                                                $endif
                                              $endif
03F1  E9 0361 R                               jmp     state_done
```

```
                        level_four_interface

                                page

                        ;
                        ;       rcv state
                        ;
                        ; Note: This routine duplicates the essence of some code from ql4_rcvmsg.
                        ;

03F4  26: 8A 44 3E      state_rcv:      mov     al,es:[si].recv_status
03F8  3C 05                             cmp     al,tf_in_prog
                                        $ifnot  e
03FC  3C 00                               cmp   al,tf_idle
                                          $if   e
0400  C7 45 0E 0002                         mov [di].qrb_status,l4st_done   ;ended ok; set status
0405  26: 8B 44 46                          mov ax,es:[si].recv_length      ;move length to our rb
0409  89 45 14                              mov [di].qrb_rcvlength,ax
040C  26: 8A 44 3F                          mov al,es:[si].recv_type        ;move type to our rb
0410  88 45 1D                              mov [di].qrb_rcvtype,al
0413  E8 04C3 R                             call post                       ;post the task
                                          $else
0419  E8 0476 R                             call abort          ;ended badly, abort
041C  E8 04C3 R                             call post
041F  E9 034D R                             jmp  search_qrbs
                                          $endif
                                        $endif
0422  E9 0361 R                         jmp     state_done


                        ;
                        ;       snd state
                        ;

0425  26: 8A 44 09      state_snd:      mov     al,es:[si].send_status
0429  3C 05                             cmp     al,tf_in_prog   ;check for not in_prog
                                        $ifnot  e
042D  26: 8B 5C 1E                        mov   bx,es:[si].hdr_dest_host;was this broadcast?
0431  26: 23 5C 20                        and   bx,es:[si].hdr_dest_host+2
0435  26: 23 5C 22                        and   bx,es:[si].hdr_dest_host+4
0439  83 FB FF                            cmp   bx,0ffffh
043C  74 13                               je    state_snd_abort                 ;yes: abort now
043E  3C 06                               cmp   al,tf_ack_wait  ;if non-broadcast, check for not ack wait
                                          $ifnot e
0442  3C 00                                     cmp al,tf_idle
                                                $if e
0446  C7 45 0E 0002                                mov  [di].qrb_status,l4st_done   ;ended ok; set status
044B  E8 04C3 R                                    call post                        ;post the task
                                                $else
0451  E8 0476 R         state_snd_abort:           call abort   ;ended badly (or broadcast), abort
0454  E8 04C3 R                                    call post
0457  E9 034D R                                    jmp  search_qrbs
                                                $endif
                                          $endif
                                        $endif
045A  E9 0361 R                         jmp     state_done
```

```
                        level_four_interface

                                page

                        ;
                        ;       disconn state
                        ;

045D  26: 8A 44 08      state_disconn:  mov     al,es:[si].conn_status
0461  3C 00                             cmp     al,cn_established
                                        $ifnot  e
0465  C7 45 0E 0000                       mov   [di].qrb_status,l4st_uncon      ;ended; set status
046A  E8 04C3 R                           call  post                            ;post the task
046D  E8 048D R                           call  free_rb                         ;and return the l4rb
0470  E9 034D R                           jmp   search_qrbs
                                        $endif
0473  E9 0361 R                         jmp     state_done

0476                    wait_int        endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       Abort the connection
                        ;
                        ; Call the L4 abort routine, then free the L4 rb.
                        ;
                        ; The real rb is in es:si
                        ; The Queen rb is in ds:di
                        ;

0476                    abort           proc    near
0476  1E                                push    ds
0477  8C C0                                mov   ax,es
0479  8E D8                                mov   ds,ax
                                           l4_call abort_conn     ;abort it, rb in ds:si
0480  E8 04DF R                           call  click
0483  1F                                pop     ds
0484  C7 45 0E 0000                    mov     [di].qrb_status,l4st_uncon      ;disconnected
0489  E8 048D R                       call    free_rb                         ;free the rb
048C  C3                              ret
048D                    abort           endp
```

```
                        level_four_interface

                                page


                        ;
                        ;       Release the L4 rb and remove the qrb from the chain
                        ;       of rbs to look for activity on.
                        ;
                        ;       es:si points to the L4 rb
                        ;       ds:di points to the qrb
                        ;
                        ;       (Be careful to not call this as a subroutine which returns to
                        ;       the wait_intr routine, which would try to step to the next qrb.
                        ;       This is a standard problem with routines that delete their own
                        ;       node from a linked list when called from another routine which
                        ;       is traversing the list.)
                        ;

048D                    free_rb         proc    near
048D  1E                                push    ds
048E  8C C0                               mov   ax,es
0490  8E D8                               mov   ds,ax
                                          l4_call release_rb    ;easy part: release the l4 rb
0497  1F                                pop     ds

0498  8B 1E 0000 R                     mov     bx,qrb_list     ;singly-linked qrb list deletion
049C  3B DF                             cmp     bx,di           ;are we the head?
                                        $if     e
04A0  8B 5F 0C                            mov   bx,[bx].qrb_l4link ;yes: special case
04A3  89 1E 0000 R                        mov   qrb_list,bx
                                        $else                   ;no: search
                                          $do
04AA  3B 7F 0C                               cmp  di,[bx].qrb_l4link    ;are we next?
                                             $if  e
04AF  8B 45 0C                                  mov  ax,[di].qrb_l4link ;yes: delink us
04B2  89 47 0C                                  mov  [bx].qrb_l4link,ax
04B5  EB 06 90                                  jmp  free_rb_exit       ;and exit
                                             $endif
04B8  8B 5F 0C                               mov bx,[bx].qrb_l4link     ;next qrb
                                          $repeat
                                        $endif
04BD  C7 45 0C 0000     free_rb_exit:   mov     [di].qrb_l4link,0       ;for neatness
04C2  C3                                ret
04C3                    free_rb         endp
```

```
                        level_four_interface

                                page

                        ;
                        ;       post
                        ;
                        ; Set the internal qrb state to "idle".
                        ; Set "interrupt" true to indicate something significant has happened and
                        ;   wait_intr() is supposed to return to it's caller.
                        ; Post the waiting task by mailing the address of the qrb to the
                        ;   mailbox whose address is in the qrb.
                        ;
                        ; Enter and exit with   &qrb in ds:di
                        ;                       &rb in es:si
                        ;

                                extrn  mail_sen:near;d
04C3                    post            proc    near
04C3  C6 45 0A 00                       mov     [di].qrb_state,st_idle          ;state is idle
04C7  2E: C6 06 0085 R 01               mov     interrupt,1     ;"interrupt" has occurred.

04CD  56                                push    si              ;save si
04CE  06                                push    es              ;save es
04CF  8C D8                             mov     ax,ds
04D1  8E C0                             mov     es,ax           ;make es=ds for lattice
04D3  57                                push    di              ;&qrb is 2nd arg
04D4  FF 75 04                          push    [di].qrb_mail   ;qrb.mail is 1st arg
04D7  E8 0000 E                         call    mail_sen;d      ;call mail_send(qrb.mail,&qrb)
04DA  5F                                pop     di              ;throw away first arg
04DB  5F                                pop     di              ;restore di from 2nd arg
04DC  07                                pop     es              ;restore es
04DD  5E                                pop     si              ;restore si

04DE  C3                                ret
04DF                    post            endp
```

```
                                level_four_interface

                                        page

                                ;
                                ;       Half-click speaker.   Destroys no registers.
                                ;

04DF                            click           proc    near
04DF  50                                        push    ax
04E0  E4 61                                     in      al,spkr_port
04E2  34 02                                     xor     al,02           ;toggle bit
04E4  24 FE                                     and     al,0feh         ;leave timer gate alone
04E6  E6 61                                     out     spkr_port,al
04E8  58                                        pop     ax
04E9  C3                                        ret
04EA                            click           endp


                                ;
                                ;       Fatal errors
                                ;

04EA  8D 16 00BB R              error_nwinit:   lea     dx,msg_nwinit
04EE  EB 12                                     jmp     short error_msg
04F0  8D 16 00D2 R              error_rb:       lea     dx,msg_rb
04F4  EB 0C                                     jmp     short error_msg
04F6  8D 16 00EB R              error_socket:   lea     dx,msg_socket
04FA  EB 06                                     jmp     short error_msg
04FC  8D 16 0100 R              err_not_idle:   lea     dx,msg_notidle
0500  EB 00                                     jmp     short error_msg

0502                            error_msg:              ;ds:dx points to string
0502  B4 09                                     mov     ah,dosint_prints
0504  CD 21                                     int     dos_int
0506  E8 0000 E                                 call    exit            ;call Lattice C exit routine

                                        endps

                                        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
DSEG	.0003
ENDDS	.0001
ENDPS	.0001
L4_CALL	.0003
PSEG	.0003

Structures and records:

Name	Width / Shift	# fields / Width	Mask	Initial
ETHER_HEADER	.002E	0015		
ARC_CODE	0000			
GARBAGE	0001			
PACKET_NUM	0002			
FRAGMENT	0003			
CHECKSUM	0004			
E_LENGTH	0006			
TRANS_CTRL	0008			
PACKET_TYPE	0009			
DEST_NETWORK	000A			
DEST_HOST	000E			
DEST_SOCKET	0014			
SRC_NETWORK	0016			
SRC_HOST	001A			
SRC_SOCKET	0020			
CONN_CTRL	0022			
DATA_TYPE	0023			

```
  SOURCE_ID. . . . . .          0024
  DEST_ID. . . . . .            0026
  SEQ_NUM. . . . . .            0028
  ACK_NUM. . . . . .            002A
  ALLOC_NUM. . . . . .          002C
HOST_ID. . . . . .0006          0001
LEVEL_4_PUBS . . . . . .004B    001D
  L4_VERSION . . . . . .        0000
  L4_FEATURES. . . . . .        0002
  L4_STATUS. . . . . .          0004
  OUR_ARC. . . . . .            0005
  L4_DEBUG . . . . . .          0006
  L4_IN_USE. . . . . .          0008
  LONG_PKT_MODE. . . . . .      0009
  OUR_ETHER. . . . . .          000A
  FREE_HEAD. . . . . .          0010
  ACTIVE_HEAD. . . . . .        0014
  D_TO_ACCEPT_WAIT . . . . .    0018
  D_CONN_TRIES . . . . . .      001A
  D_TO_ACK_WAIT. . . . . .      001C
  D_MESSAGE_TRIES. . . . . .    001E
  D_TO_PKT_WAIT. . . . . .      0020
  TO_PKT_KEEP. . . . . .        0022
  TO_TA_WAIT . . . . . .        0024
  OLD_DISKIO_VECTOR. . . . .    0026
  ABORT_VECTOR . . . . . .      002A
  USER_EXIT_VECTOR . . . . .    002E
  NIC_SEG. . . . . .            0032
  PUB_SPARE1 . . . . . .        0034
  PUB_SPARE2 . . . . . .        0036
  PUB_SPARE3 . . . . . .        0038
  PUB_SPARE4 . . . . . .        003A
  OLD_BOOT_VECTOR. . . . . .    003C
  BOOT_HOST. . . . . .          0040
  BOOT_RB. . . . . .            0046
  BOOTED_FROM_NET. . . . . .    004A
QRB. . . . . .0021              0011
  QRB_ID . . . . . .            0000
  QRB_MLINK. . . . . .          0002
  QRB_MAIL . . . . . .          0004
  QRB_RB . . . . . .            0006
  QRB_STATE. . . . . .          000A
  QRB_L4LINK . . . . . .        000C
  QRB_STATUS . . . . . .        000E
  QRB_CHURNCNT . . . . . .      0010
  QRB_RCVPTR . . . . . .        0012
  QRB_RCVLENGTH. . . . . .      0014
  QRB_RCVLIMIT . . . . . .      0016
  QRB_SNDPTR . . . . . .        0018
  QRB_SNDLENGTH. . . . . .      001A
  QRB_SNDTYPE. . . . . .        001C
  QRB_RCVTYPE. . . . . .        001D
  QRB_WKS. . . . . .            001E
  QRB_PKTHDR . . . . . .        0020
```

```
REQ_BLOCK. . . . . .0086    0043
  NEXT . . . . . .          0000
  RB_SIG . . . . . .        0004
  PROTOCOL_MODE. . . . . .  0006
  RB_IN_USE. . . . . .      0007
  CONN_STATUS. . . . . .    0008
  SEND_STATUS. . . . . .    0009
  SEND_PTR . . . . . .      000A
  SEND_LENGTH. . . . . .    000E
  HDR_ARC_CODE . . . . . .  0010
  HDR_GARBAGE. . . . . .    0011
  HDR_PACKET_NUM . . . . .  0012
  HDR_FRAGMENT . . . . . .  0013
  HDR_CHECKSUM . . . . . .  0014
  HDR_E_LENGTH . . . . . .  0016
  HDR_TRANS_CTRL . . . . .  0018
  HDR_PACKET_TYPE. . . . .  0019
  HDR_DEST_NETWORK . . . .  001A
  HDR_DEST_HOST. . . . . .  001E
  HDR_DEST_SOCKET. . . . .  0024
  HDR_SRC_NETWORK. . . . .  0026
  HDR_SRC_HOST . . . . . .  002A
  HDR_SRC_SOCKET . . . . .  0030
  HDR_CONN_CTRL. . . . . .  0032
  HDR_DATA_TYPE. . . . . .  0033
  HDR_SOURCE_ID. . . . . .  0034
  HDR_DEST_ID. . . . . .    0036
  HDR_SEQ_NUM. . . . . .    0038
  HDR_ACK_NUM. . . . . .    003A
  HDR_ALLOC_NUM. . . . . .  003C
  RECV_STATUS. . . . . .    003E
  RECV_TYPE. . . . . .      003F
  RECV_PTR . . . . . .      0040
  RECV_LIMIT . . . . . .    0044
  RECV_LENGTH. . . . . .    0046
  RB_TO_ACCEPT_WAIT. . . .  0048
  RB_CONN_TRIES. . . . . .  004A
  RB_TO_ACK_WAIT . . . . .  004C
  RB_MESSAGE_TRIES . . . .  004E
  RB_TO_PKT_WAIT . . . . .  0050
  PEND_VALID . . . . . .    0052
  PEND_TYPE. . . . . .      0053
  HIS_ARC. . . . . .        0054
  HIS_BCST . . . . . .      0055
  RB_SPARE1. . . . . .      0056
  RB_SPARE2. . . . . .      0058
  RB_SPARE3. . . . . .      005A
  RB_SPARE4. . . . . .      005C
  CONN_STATE . . . . . .    005E
  SEND_STATE . . . . . .    005F
  SEND_CHANGED . . . . . .  0060
  SEND_CURSOR. . . . . .    0062
  SEND_REMAINING . . . . .  0066
  OUR_ACK_REQ. . . . . .    0068
```

```
  OUR_REQ_VALID. . . . . .       006A
  RECV_STATE . . . . . .         006B
  RECV_CHANGED . . . . . .       006C
  RECV_CURSOR. . . . . .         006E
  HIS_SEQ. . . . . .             0072
  HIS_ACK. . . . . .             0074
  HIS_ALLOC. . . . . .           0076
  PEND_BUF . . . . . .           0078
  PEND_HDR . . . . . .           007A
  PEND_START . . . . . .         007E
  SEND_RETRIES . . . . . .       0080
  MESS_START_SEQ . . . . . .     0082
  ACK_FLAGS. . . . . .           0084
  L2_SEQ . . . . . .             0085
TIME_OUT_REC . . . . . .000A     0005
  TO_ACCEPT_WAIT . . . . . .     0000
  CONN_TRIES . . . . . .         0002
  TO_ACK_WAIT. . . . . .         0004
  MESSAGE_TRIES. . . . . .       0006
  TO_PKT_WAIT. . . . . .         0008

Segments and Groups:

                N a m e                 Size    Align   Combine Class

DGROUP . . . . . .GROUP
  DATA . . . . . .               0002    WORD    PUBLIC  'DATA'
NIC_SEGMENT. . . . . .100E       PARA    NONE
PGROUP . . . . . .GROUP
  PROG . . . . . .               0509    BYTE    PUBLIC  'PROG'

Symbols:

                N a m e                 Type    Value   Attr

ABORT. . . . . .N PROC  0476     PROG    Length =0017
ADDR_ALL . . . . . .Number  FFFF
AL4LOCATE. . . . . .N PROC  0038     PROG    Length =0027
BAD_NIC. . . . . .Number  0002
BAD_RAM. . . . . .Number  0004
BAD_RIM. . . . . .Number  0003
BAD_SOCK . . . . . .Number  0004
BINGO. . . . . .L NEAR  008C     PROG
BUILD_LOOP . . . . . .L NEAR  009C     PROG
BUILD_TABLE. . . . . .N PROC  0091     PROG    Length =0024
CHURN. . . . . .L NEAR  0321     PROG
CLICK. . . . . .N PROC  04DF     PROG    Length =0008
CN_ACCEPT_WAIT . . . . . .Number  0005
CN_ESTABLISHED . . . . . .Number  0000
CN_FAIL. . . . . .Number  0002
CN_NOT_CONN. . . . . .Number  0001
CN_OPEN_RCVD . . . . . .Number  0006
CN_PARM_ERROR. . . . . .Number  0004
CN_STATE_ERROR . . . . . .Number  0003
```

```
COM. . . . . .Number  0000
CR . . . . . .Number  000D
D8086. . . . . .Number  0000
DEBUG. . . . . .L NEAR  0000    PROG    External
DISKIO . . . . . .Number  0013
DOSINT_PRINTS. . . . . .Number  0009
DOS_INT. . . . . .Number  0021
ERROR_MSG. . . . . .L NEAR  0502    PROG
ERROR_NWINIT . . . . . .L NEAR  04EA    PROG
ERROR_RB . . . . . .L NEAR  04F0    PROG
ERROR_SOCKET . . . . . .L NEAR  04F6    PROG
ERR_NOT_IDLE . . . . . .L NEAR  04FC    PROG
ESC. . . . . .Number  001B
EXIT . . . . . .L NEAR  0000    PROG    External
FALSE. . . . . .Number  0000
FIND_LOOP. . . . . .L NEAR  0062    PROG
FIND_NIC . . . . . .N PROC  005F    PROG    Length =0032
FIND_NIC_EXIT. . . . . .L NEAR  0090    PROG
FREE_RB. . . . . .N PROC  048D    PROG    Length =0036
FREE_RB_EXIT . . . . . .L NEAR  04BD    PROG
GOOD_RB_SIG. . . . . .Number  6272
GOT_INCOMING . . . . . .L NEAR  03A3    PROG
IBM_SIG. . . . . .L WORD  1000    NIC_SEGMENT
IF$1002. . . . . .L NEAR  045A    PROG
IF$102 . . . . . .L NEAR  015E    PROG
IF$1052. . . . . .L NEAR  045A    PROG
IF$1100. . . . . .L NEAR  045A    PROG
IF$1102. . . . . .L NEAR  0451    PROG
IF$1152. . . . . .L NEAR  0473    PROG
IF$1200. . . . . .L NEAR  04BD    PROG
IF$1202. . . . . .L NEAR  04AA    PROG
IF$1250. . . . . .L NEAR  04AA    PROG
IF$1302. . . . . .L NEAR  04B8    PROG
IF$152 . . . . . .L NEAR  0185    PROG
IF$202 . . . . . .L NEAR  01C0    PROG
IF$252 . . . . . .L NEAR  023A    PROG
IF$302 . . . . . .L NEAR  029B    PROG
IF$350 . . . . . .L NEAR  029B    PROG
IF$352 . . . . . .L NEAR  0291    PROG
IF$402 . . . . . .L NEAR  02AE    PROG
IF$452 . . . . . .L NEAR  02F8    PROG
IF$502 . . . . . .L NEAR  0346    PROG
IF$52. . . . . .L NEAR  0134    PROG
IF$552 . . . . . .L NEAR  0346    PROG
IF$602 . . . . . .L NEAR  0346    PROG
IF$650 . . . . . .L NEAR  0351    PROG
IF$702 . . . . . .L NEAR  0378    PROG
IF$752 . . . . . .L NEAR  03A0    PROG
IF$802 . . . . . .L NEAR  03F1    PROG
IF$850 . . . . . .L NEAR  03F1    PROG
IF$852 . . . . . .L NEAR  03E8    PROG
IF$902 . . . . . .L NEAR  0422    PROG
IF$950 . . . . . .L NEAR  0422    PROG
IF$952 . . . . . .L NEAR  0419    PROG
```

```
IF$L . . . . . .Number  0000
IF$N . . . . . .Number  0514
IF$NS. . . . . .Number  04B0
IF$NS1 . . . . . .Number  04B0
IF$NS2 . . . . . .Number  04E2
IF$NS3 . . . . . .Number  0514
IF$T . . . . . .Number  0003
IF$T1. . . . . .Number  0003
IF$T2. . . . . .Number  0000
IF$T3. . . . . .Number  0002
INFINITY . . . . . .Number  FFFF
INTERRUPT. . . . . .L BYTE  00B5    PROG
JMP_INS. . . . . .L BYTE  1003    NIC_SEGMENT     Length =0003
JUMP_TABLE . . . . . .L WORD  0315    PROG
KEYBOARD . . . . . .Number  0016
KEY_COUNT. . . . . .L BYTE  00B6    PROG
L4ST_BUSY. . . . . .Number  0001
L4ST_DONE. . . . . .Number  0002
L4ST_FAILED. . . . . .Number  0004
L4ST_PARTIAL . . . . . .Number  0003
L4ST_UNCON . . . . . .Number  0000
L4_ENTRIES . . . . . .L DWORD 0000    PROG    Length =000E
L4_INSTALLED . . . . . .L NEAR  005B    PROG
L4_IN_OUR_SEG. . . . . .Number  0000
L4_IN_ROM. . . . . .L NEAR  0049    PROG
L4_LOC_EXIT. . . . . .L NEAR  005E    PROG
L4_OK. . . . . .Number  0000
L8086. . . . . .Number  0000
LDATA. . . . . .Number  0000
LF . . . . . .Number  000A
LPROG. . . . . .Number  0000
MAIL_SEN . . . . . .L NEAR  0000    PROG    External
MAX_ENTRY. . . . . .Alias   OFF_L4GET_PTRS
MSDOS. . . . . .Number  0002
MSG_NOTIDLE. . . . . .L BYTE  0100    PROG
MSG_NWINIT . . . . . .L BYTE  00BB    PROG
MSG_RB . . . . . .L BYTE  00D2    PROG
MSG_SOCKET . . . . . .L BYTE  00EB    PROG
NEXT_2K. . . . . .L NEAR  0081    PROG
NIC_RAM. . . . . .L BYTE  0000    NIC_SEGMENT     Length =1000
NIL. . . . . .Number  0000
NO_L4. . . . . .Number  0005
NO_NIC . . . . . .Number  0001
OFF_ABORT_CONN . . . . . .Number  0009
OFF_ACK_NOW. . . . . .Number  0006
OFF_ACTIVATE_RB. . . . . .Number  0001
OFF_CONNECT. . . . . .Number  0007
OFF_DISCONN. . . . . .Number  0008
OFF_IGNORE . . . . . .Number  0004
OFF_L4CHURN. . . . . .Number  000C
OFF_L4GET_PTRS . . . . . .Number  000D
OFF_L4INIT . . . . . .Number  0000
OFF_LISTEN . . . . . .Number  0003
OFF_OPEN_RECV. . . . . .Number  0005
```

```
OFF_RECV_MSG . . . . . .Number  000B
OFF_RELEASE_RB . . . . . .Number  0002
OFF_SEND_MSG . . . . . .Number  000A
ONE_SECOND . . . . . .Number  0012
ON_NIC . . . . . .Number  0000
OURWKS1. . . . . .L WORD  00B7    PROG
OURWKS2. . . . . .L WORD  00B9    PROG
P8086. . . . . .Number  0000
POST . . . . . .N PROC  04C3    PROG    Length =001C
QL4_ABOR . . . . . .N PROC  0160    PROG    Global  Length =000E
QL4_CONN . . . . . .N PROC  01A9    PROG    Global  Length =0081
QL4_DISC . . . . . .N PROC  02E9    PROG    Global  Length =002C
QL4_INIT . . . . . .N PROC  0126    PROG    Global  Length =000F
QL4_LIST . . . . . .N PROC  0135    PROG    Global  Length =002B
QL4_OPEN . . . . . .N PROC  016E    PROG    Global  Length =003B
QL4_RCVM . . . . . .N PROC  022A    PROG    Global  Length =0074
QL4_SNDM . . . . . .N PROC  029E    PROG    Global  Length =004B
QRB_LIST . . . . . .L WORD  0000    DATA    Global
READY_TC . . . . . .V WORD  0000    DATA    External
ROM_EADDR. . . . . .L WORD  1006    NIC_SEGMENT     Length =0003
ROM_ENTRIES. . . . . .L WORD  100C    NIC_SEGMENT
ROM_LENGTH . . . . . .L BYTE  1002    NIC_SEGMENT
S8086. . . . . .Number  0001
SEARCH_QRBS. . . . . .L NEAR  034D    PROG
SOCK_IN_USE. . . . . .Number  0003
SOCK_NOT_FOUND . . . . . .Number  0001
SOCK_OK. . . . . .Number  0000
SPKR_PORT. . . . . .Number  0061
STATE_DISCONN. . . . . .L NEAR  045D    PROG
STATE_DONE . . . . . .L NEAR  0361    PROG
STATE_OPENRCV. . . . . .L NEAR  0379    PROG
STATE_RCV. . . . . .L NEAR  03F4    PROG
STATE_SENDACK. . . . . .L NEAR  03D1    PROG
STATE_SND. . . . . .L NEAR  0425    PROG
STATE_SND_ABORT. . . . . .L NEAR  0451    PROG
ST_DISCON. . . . . .Number  0008
ST_IDLE. . . . . .Number  0000
ST_OPENRCV . . . . . .Number  0002
ST_RCV . . . . . .Number  0004
ST_SENDACK . . . . . .Number  000A
ST_SND . . . . . .Number  0006
TF_ACK_WAIT. . . . . .Number  0006
TF_FAIL. . . . . .Number  0002
TF_IDLE. . . . . .Number  0000
TF_IN_PROG . . . . . .Number  0005
TF_NOT_CONN. . . . . .Number  0001
TF_NOT_IMPL. . . . . .Number  0008
TF_PARM_ERROR. . . . . .Number  0004
TF_RECV_OVFL . . . . . .Number  0007
TF_STATE_ERROR . . . . . .Number  0003
TOO_MANY_SOCKS . . . . . .Number  0002
TRUE . . . . . .Number  0001
TR_L4CHURN . . . . . .Number  0023
VIDEO. . . . . .Number  0010
```

```
WAIT_INT . . . . . .N PROC  0321    PROG    Global  Length =0155
XNS_ECHO . . . . . .Number  0003
XNS_ERROR. . . . . .Number  0002
XNS_NONE . . . . . .Number  0000
XNS_PEP. . . . . .Number  0005
XNS_RIP. . . . . .Number  0001
XNS_SPP. . . . . .Number  0004

24740 Bytes free

Warning Severe
Errors  Errors
0       0
```

```
ERR  LINE  ADDR
        1                                       ttl     enable() and disable() routines
        2
        3                               ;
        4                               ;  This is a small part of what is usually in execasm.src.
        5                               ;  It is for small test programs which need disable/enable and
        6                               ;  not the rest of the executive.
        7                               ;
        8
        9
       10                               ;
       11                               ;       flags = enable()
       12                               ;       disable(flags)
       13                               ;
       14                               ;  Enable and disable hardware interrupts.
       15                               ;  See the commentary in the task dispatcher for the use of these routines.
       16                               ;  this must be in the supervisor mode for enable and disable interrupts to
       17                               ;  work.
       18                               ;  for now, we assume it will be in the supervisor mode.
       19                               ;
       20
       21                                       xdef    .disable
       22
       23                               .disable
       24 00000000 7000                        moveq   #0,d0           ;clear a reg
       25 00000002 40C0                        move    sr,d0           ;get current int setting ready for retur
       26 00000004 2200                        move.l  d0,d1           ;get a copy
       27 00000006 0041 0700                   ori.w   #$0700,d1       ;turn off all interrupts in copy
       28 0000000A 46C1                        move    d1,sr           ;put copy in sr
       29 0000000C 4E75                        rts                     ;return flags
       30
       31                                       xdef    .enable
       32
       33                               .enable
       34 0000000E 46EF 0006                   move    6(sp),sr        ;restore flags with data provided
       35 00000012 4E75                        rts
       36
       37 00000014                             end

     0 Errors,     0 Warnings
```

SYMBOL TABLE

NARG 00000000 .DISABLE R ?????00:00000000 .ENABLE R ?????00:0000000E