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

Sat 15-Feb-1986 16:59:07

Print request number 809

Station: $36

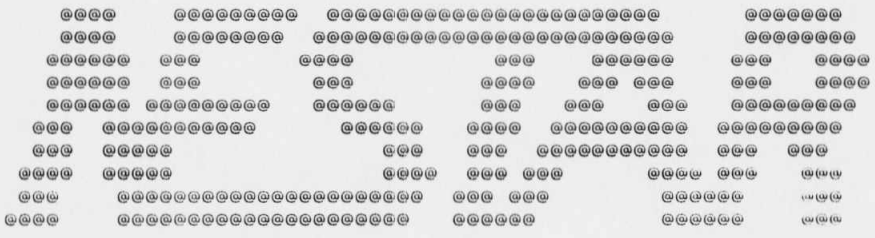
Name: L J Shustek

File Server: BUTLER ($FE)

NFS Pathname:

Filename (s):

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



- Don't truncate names when printed in window (or at least 16 or 20)

```
                page    58,132
                title   TALK - Multiple station conversation program

;-----------------------------------------------------------------------------
;
;
;   TALK        Multiple station conversation program
;
;
;       This is a background exit-and-stay-resident program which allows
;       multiple stations to engage in a simultaneous conversation.
;
;       Screen display is within four windows:  one for the typing buffer,
;       one for the display script of messages from all stations, one
;       for a list of known stations, and one for a list of known groups.
;
;       The destination of messages typed is shown by highlighting the
;       group name and/or station names in the appropriate windows.
;       The cursor arrow keys are used to change the destination.
;
;       The windows can be displayed by a keyboard trigger character and,
;       if desired, automatically when a message is received.  If the
;       automatic window popup is not chosen, any messages received are
;       put in the hidden window until displayed.
;
;
;       For build information about talk, see the file talkh.asm.
;
;
;                   (C) Copyright 1984, Nestar Systems Inc.
;
;-------------------------------------------------------------------------------
;
; Change log
;
;  When    Vers      Who           What changed
; ------- ----   ------------     -------------------
; 7/11/84        L. Shustek       Experiments.
; 7/19/84        L. Shustek       Initial internal release.
; 7/25/84        L. Shustek       Restructure into exit-and-stay-resident program.
;
; 8/ 2/84 0.32   L. Shustek       Add stn flags and better station removal logic.
;
; 8/ 7/84 0.34   L. Shustek       Change timer interrupt hook to use hw int instead
;                                   of dos user exit.  Add bold stn names.
;
; 8/23/84 0.35   L. Shustek       Higher pitch for new-station tweedle.
;                                 Implement "int kb" function 99 to return status.
;                                 Allow trigger character to close window.
;                                 Add name and version to window border.
;                                 Use short L2 timeouts only while sending msgs.
;                                 Do open_recv only every few ticks, for lower overhead.
;                                 Allow task activation during kb_int, so it can
```

```
;                                       be useful during NWCMDS, etc.
;                                       Filter out bells and linefeeds from messages.
;
; 8/28/84 0.40  L. Shustek      Major changes for addition of groups in the 4th
;                               window, and selected destinations.
;
; 8/31/84 0.41  L. Shustek      Do acknow during rcv for multi-packet connects.
;                               Remember unused groups.  Remember previous grp/stn
;                               destination.  Use newer stn name, if any.  Don't
;                               echo in display window if send-to-1-stn fails.
;                               Don't show rumors.  Low beep if send failure.
;                               Minimize group/stn window glitches by overwriting
;                               instead of clearing and rewriting when possible.
;                               Implement F1/F3 functions to copy previous text.
;                               Add kb_int function call to send a message
;                               from a program.
;
; 9/06/84 0.42  L. Shustek      Check for unused groups more often.
;                               Fix task conflicts in updating data structures.
;                               Add internal error detection.
;                               Put back rumored station display, but
;                               send silent probe messages to verify rumors.
;                               Break up into multiple assemblies, fighting
;                               with the linker to order things properly.
;
; 9/18/84 0.43  L. Shustek      Use new window package to provide a second
;                               page, and do PGUP, PGDN, HOME, END keys.
;                               Underline stations being sent to.
;                               Check user/group names for length & bad chars.
;                               Make tasks start up more cleanly after install.
;                               Allow recursive timer interrupts for tweedles.
;
; 9/21/84 0.44  L.Shustek       Don't acknowledge (l4_acknow) broadcast msgs.
;                               Don't probe rumors if fileserver connection is
;                               in progress;  just get out as quickly as possible.
;
; 9/21/84 0.45  L. Shustek      Fix rare <iu> err caused by check_incoming not
;                               looking at L4_busy if open_recv was already true.
;
; 2/22/85 0.46  L. Shustek      Fix RB chain end test in busy_rb_check
;                               for compatibility with L4 roms after v1.1.
;
; 6/19/85 0.47  L. Shustek      Add int16h function codes to suspend and resume
;                               interrupt processing.  For coexistence with CARDCHK.
;                               Change "Group:" to "Groups:"; add "Users:".
;
; 2/14/86 0.48  L. Shustek      Expand user window to 3 lines; widen all windows.
;                               Don't do broadcast until all system RBs are quiescent;
;                                 fixes occasional 30 second delay at installation due
;                                 to fileserver bug: broadcast causes the current
;                                 connection to be aborted!
;                               Print install message before setting interrupt vectors.
;                               Don't open the windows if screen is in graphic mode.
;                               Add TIMESTAMP keyword to enable msg timestamps.
```

```
;                                       Add POPDOWN mode to auto-popdown window after n seconds.
;                                       Add CHANNEL to specify separate TALK channels.
;                                       Allow color attribute modifier for better colors.
;                                       Do an END operation when the window is opened.
;                                       Change number of display pages from 2 to 3.
;                                       Allow user window to be specified at invocation.
;
;-----------------------------------------------------------------------------------------
;
;  Yet to do (or think about):
;
;       Allow a way to turn it off, or at least stop the bells.
;       User-exit to supplied procedure for certain received message types.
;       Ask for name if no USER= environment string is found?
;       "TALK ?" should give help info from init phase
;       Instead of "already installed", parse command line and
;          reset options;  show new options.
;       Wait for retrace on graphics screens?  (Slow!)  (Actually is a
;          feature request for the WINDOW package, not TALK.)
;       Bug: stn $FF should be ok, but we use it for a "broadcast" flag in
;         the sendmsg routine.
;       Add the remaining DOS editing key functions.
;       Add a priority message type, or a "super popup" mode for important
;          messages.
;       TALK ON and TALK OFF commands?  (Why?)
;       Adjust the user (and group?) window dynamically depending on the
;          number of users.
;       Bug: Nulls are getting into the group list somehow.
;
;-----------------------------------------------------------------------------------------


;-----------------------------------------------------------------------------------------
;
;
;               A short course on the internal design of TALK
;
;
; 1. Gereral Structure
;
;
; TALK consists of a resident part, which becomes part of DOS using the
; exit-and-stay-resident function call, and a transient part which is used
; for initialization.  The resident part contains interrupt routines, a
; task dispatcher, and 4 tasks for processing concurrent operations.
; The static structure looks roughly as follows:
;
;       TALK data area
;       Main Tasks
;           Keyboard Task
;           Receive Task
;           Transmit Task
;       List and Utility routines
;       Dispatcher
```

```
;       Interrupt routines
;           Interrupt Task
;           Keyboard interrupt (int 16h) routine
;           L4 Exit interrupt routine
;           Timer interrupt routine
;       Shutdown procedure
;       Initialization
;           Command parsing
;           Window initialization
;           Network initialization
;           Evironment string parsing
;           Interrupt routine installation
;           Exit-and-stay-resident
;
;
; 2. Task dispatching
;
;
; The dispatcher is a round-robin non-preemptive scheduler which just
; saves and restores registers on a stack for each task.  All task code is
; required to include a "call dispatch" in every loop which takes an
; indeterminate amount of time.
;
; The task dispatcher is started from the Interrupt Task, which is in turn
; called from one of the interrupt routines when a message must be processed
; in either background or foreground.  The only difference between background
; and foreground is whether the windows are displayed, and that is controlled
; by the Keyboard Task on the basis of flags set by the interrupt routines.
; Once started, the tasks will run until the windows are closed and no messages
; are in progress;  state flags are set by the various tasks which are used
; by the Interrupt Task to indicate when it should return to its interrupt
; routine and thus suspend the task dispatcher until the next time a signficant
; event has occurred.
;
;
; 3. Message formats
;
;
; TALK uses only one well-known socket.  Within that socket, three message
; types are used, for:
;
;       genl_msg_type:    General messages between stations
;       logon_msg_type:   Logon broadcast messages
;       logoff_msg_type:  Logoff messages (not currently sent)
;
; Within each message, regardless of type, may be several submessages.
; Each submessage starts with a 1-byte code and a variable-length data part.
; The submessages are:
;
;       grp_subtype:  the complete group namelist known by the sender.
;       stn_subtype:  the complete station namelist known by the sender.
;       txt_subtype:  a text message to be displayed, terminated with CR.
;
; Currently all three submessages are included as part of ANY messages sent
```

```
; between stations, except that the Logon broadcast message does not have
; any txt_subtype.
;
; When a station first installs TALK, a broadcast message is sent.  All
; stations which hear the broadcast send a message containing group and
; station lists, from which the receiving station constructs its initial
; lists.  From then on, every message sent or received contains group and
; station lists along with the text.
;
; Group information is always taken verbatim.  Station information which is
; secondhand is treated as a "rumor", which is to say that the existence of
; a station is not confirmed until a message is received from or successfully
; sent to that station.  A station is removed from the station list whenever
; a direct send to it fails.  (All transmissions are retried "retries" times.)
; Rumored stations may or may not be displayed, depending on the setting of the
; assembly switch "show_rumors".
;
;
; 4. Task interaction
;
;
; As in all multitasking systems, care must be taken when accessing shared
; data structures.  The dispatcher in TALK has the property that tasks are
; switched only when an explicit call to DISPATCH is made, so critical sections
; that update shared data need not generally lock access to the data.
;
; The real danger comes in forgetting which procedures call DISPATCH.  In
; particular, you must always remember that sending or receiving messages
; does so, and that when other tasks get control, data like the station lists
; and group lists may change.  For example, the send task walks down the
; station-list to send messages to each station, and so the receive task
; must not delete any station-list records if the send task is busy.
;
; The following is a list of places where the tasks do a DISPATCH.  Please keep
; this list up-to-date so that shared data access can be monitored:
;
; keyboard_task:- Wait for other tasks to idle before shutdown
;                 - Shutdown (the DISPATCH call which stops the dispatcher)
;                 - Wait for keyboard character
;                 - Wait for send_task to release keyboard buffer
;
; receive_task: - Wait for send_task to do a broadcast
;                 - Receive message
;
; send_task:    - Wait for work to do (any of 4 or so conditions)
;                 - Send a message
;
;
;
;...more words to come, as I have the patience...
;
;-------------------------------------------------------------------------------
;-------------------------------------------------------------------------------
```

```
                                .sall           ;suppress macro expansions

                                subttl  ...Permanently resident code...

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

                        C       include talkv.asm               ;symbols and variables
                        C
                        C       ;-----------------------------------------------------------------------------
                        C       ;
                        C       ;  talkv.asm    TALK symbols and variables
                        C       ;
                        C       ;-----------------------------------------------------------------------------
                        C
                        C       version         macro
                        C                       db      '0.48'                  ;version number
                        C                       endm
                        C
= 0D04                  C       talk_wks        equ     0d04h ;+channel!;well-known socket base.  Change it
                        C                                                       ;whenver packet format changes'
                        C
                        C                       .lfcond
= FFFF                  C       c_true          equ     0ffffh
= 0000                  C       c_false         equ     0000h
                        C
=                       C       show_rumors     equ     c_true                  ;show rumored stns in stn_list?
=                       C       assert          equ     c_true                  ;generate internal error checking?
                        C
                        C
                        C                       extrn   window:near
                        C
                        C       cgrp            group   hseg,vseg,cseg
                        C
0000                    C       hseg            segment common 'hsegcl' ;dummy (this assembler'll kill me yet)
0000                    C       hseg            ends
                        C
0000                    C       vseg            segment common 'vsegcl'
                        C
                        C       ;
                        C       ;               LEVEL 4 SYMBOLS
                        C       ;
                        C
= 0000                  C       on_nic          equ     0
= 0000                  C       l4_in_our_seg   equ     0
                        C
                        C       ;;              include e:l4asm.itf             ;level 4 interface
                        C                       .list
                        C
                        C
                        C       ;
                        C       ;               SYMBOLS
                        C       ;
```

```
                        level_four_interface

                        C
= 0004                  C  ntasks          equ     4                    ;number of tasks
= 0258                  C  max_stn_list    equ     600                  ;max size of station list
= 00C8                  C  max_grp_list    equ     200                  ;max size of group list
= 0010                  C  max_groups      equ     16                   ;max number of groups (<=16)
= 0010                  C  max_namesize    equ     16                   ;largest name allowed
                        C                                               ; (for internal debug only)
= 0003                  C  n_pages         equ     3                    ;number of display pages
                        C
= 0003                  C  how_often       equ     3                    ;3 x 55 = 165 msec check for msgs
= 0006                  C  timeout         equ     6                    ;6 x 55 = 330 msec timeouts
= 0002                  C  retries         equ     2                    ;number of retries
                        C
= 0097                  C  logoff_msg_type equ     97h                  ;logoff message type
= 0098                  C  logon_msg_type  equ     98h                  ;logon message
= 0099                  C  genl_msg_type   equ     99h                  ;talk general message type
= 0096                  C  rqscrn_msg_type equ     96h                  ;requestscreen message type
= 0095                  C  scrn_msg_type   equ     95h                  ;screen message type
= 0094                  C  kbchar_msg_type equ     94h                  ;kb char message type
                        C
= 00AA                  C  stn_subtype     equ     0aah                 ;subtype for station recs
= 00BB                  C  grp_subtype     equ     0bbh                 ;subtype for groups recs
= 00CC                  C  txt_subtype     equ     0cch                 ;subtype for text
                        C
                        C
= 0016                  C  key_int         equ     16h                  ;rom bios keyboard int
= 0000                  C  key_read        equ     0                    ;read key function
= 0001                  C  key_stat        equ     1                    ;key status function
= 0002                  C  key_shift       equ     2                    ;key shift status func.
= 0063                  C  key_talk        equ     99                   ;nestar extended info for talk
= 0062                  C  key_nwcmds      equ     98                   ;nestar extended info for nwcmds
= 0061                  C  key_talk_send   equ     97                   ;nestar extended cmd for send_msg
= 0060                  C  key_talk_stop   equ     96                   ;nestar extended cmd for "suspend TALK"
= 005F                  C  key_talk_start  equ     95                   ;nestar extended cmd for "resume TALK"
                        C
                        C  ; Nestar extended int 16h functions return AH negated to indicate that they
                        C  ; did something;  if the program (eg TALK) is not installed then the rom bios
                        C  ; will return with AH unchanged.
                        C  ;
                        C  ; The extended-info call (key_talk, key_nwcmds, etc.) returns the following
                        C  ; flags in AL, or'd by all copies of the program:
                        C
= 0001                  C  key_ext_active  equ     01                   ;extended info flag: active
= 0002                  C  key_ext_idle    equ     02                   ;extended info flag: idle
                        C
                        C
= 0011                  C  equip_int       equ     11h                  ;equipment determination int
                        C
= 001A                  C  timer_int       equ     1ah                  ;rom bios timer int
= 0000                  C  timer_read      equ     0                    ;read timer into cx:dx
                        C
= 0010                  C  video_int       equ     10h                  ;rom bios video int
= 0002                  C  video_setcursor equ     2                    ;set cursor from dx
= 0003                  C  video_getcursor equ     3                    ;get cursor in dx
```

```
                        level_four_interface

= 0006                  C  video_scroll     equ     6               ;scroll or clear screen
= 000A                  C  video_write_ch   equ     10              ;write single character
= 000E                  C  video_write_tty  equ     14              ;write tty function
                        C
                        C
= 0021                  C  dos_int          equ     21h             ;dos function call interrupt
                        C
= 0002                  C  dosint_printc    equ     02h             ;print character in dl
= 0009                  C  dosint_prints    equ     09h             ;print string at ds:dx until '$'
= 0031                  C  dosint_exitstay  equ     31h             ;exit but stay resident
= 004C                  C  dosint_exit      equ     4ch             ;exit program normally
                        C
= 002C                  C  psp_environ      equ     2ch             ;offset of env segment in psp
                        C
                        C
= 0008                  C  timer_tick       equ     08h             ;timer tick hardware interrupt
= 0043                  C  timer_ctrl       equ     043h            ;timer (8253) control register
= 00A6                  C  timer_ctrl_msb   equ     0a6h            ;  load msb of counter 2
                        C
= 0042                  C  timer_ch2        equ     042h            ;timer (8253) channel 2 data
= 0061                  C  kb_ctrl          equ     061h            ;keyboard (8255) control: 02h is spkr
= 0003                  C  kb_ctrl_spkr     equ     003h            ;   speaker enable bits
                        C
                        C
= 0007                  C  bell             equ     07h             ;bell
= 0008                  C  bs               equ     08h             ;backspace
= 000A                  C  lf               equ     0ah             ;linefeed
= 000D                  C  cr               equ     0dh             ;carriage return
= 001B                  C  esc              equ     1bh             ;escape
                        C
= 4800                  C  key_up           equ     4800h           ;keyboard arrow keys
= 5000                  C  key_down         equ     5000h
= 4B00                  C  key_left         equ     4b00h
= 4D00                  C  key_right        equ     4d00h
                        C
= 3B00                  C  key_f1           equ     3b00h           ;keyboard F1 key
= 3D00                  C  key_f3           equ     3d00h           ;keyboard F3 key
                        C
= 5200                  C  key_ins          equ     5200h           ;keyboard INS key
= 5300                  C  key_del          equ     5300h           ;keyboard DEL key
                        C
= 4700                  C  key_home         equ     4700h           ;keyboard home key
= 4900                  C  key_pgup         equ     4900h           ;keyboard pageup key
= 4F00                  C  key_end          equ     4f00h           ;keyboard end key
= 5100                  C  key_pgdn         equ     5100h           ;keyboard pagedown key
                        C
                        C
= 0000                  C  window_define    equ     0               ;window define
= 0001                  C  window_open      equ     1               ;window open
= 0002                  C  window_close     equ     2               ;window close
= 0003                  C  window_print     equ     3               ;window print character
= 0004                  C  window_getcurs   equ     4               ;window get cursor
= 0005                  C  window_setcurs   equ     5               ;window set cursor
= 0006                  C  window_select    equ     6               ;window select
```

```
                        level_four_interface

= 0007                  C  window_init     equ   7                        ;window initialization
= 0008                  C  window_setattr  equ   8                        ;window set char attribute
= 0009                  C  window_pageop   equ   9                        ;window page operations
                        C
= 0601                  C  select_dwin     equ   window_select*256+1       ;select display window
= 0602                  C  select_twin     equ   window_select*256+2       ;select typing window
= 0603                  C  select_swin     equ   window_select*256+3       ;select station window
= 0604                  C  select_gwin     equ   window_select*256+4       ;select group window
                        C
                        C
                        C  ;
                        C  ;               LOCAL STORAGE
                        C  ;
                        C
0000  03 14 15 54 41 4C C                  db    3h,14h,15h,'TALK'         ;debug marker (pi)
      4B                C
                        C
                        C
                        C  ;       Tasks
                        C
                        C
= 0002                  C  tcbsize         equ   2                         ;size of tcb
0007                    C  tcbs            label word                      ;task control blocks (TCB)
0007  0F00 R            C  task0           dw    offset cgrp:stack0
0009  1158 R            C  task1           dw    offset cgrp:stack1
000B  13B0 R            C  task2           dw    offset cgrp:stack2
000D  1608 R            C  task3           dw    offset cgrp:stack3
                        C
000F  0000              C  curtask         dw    0                         ;current TCB offset
                        C
0011  ????              C  savss           dw    ?                         ;keyboard int stack
0013  ????              C  savsp           dw    ?
                        C
                        C
                        C  ;       State flags.    1 is affirmative, 0 is negative.
                        C
                        C
0015  53 54 41 54 45 53 C                  db    'STATES'                  ;debug marker
001B  00                C  tasks_active    db    0                         ;task dispatcher active
001C  00                C  send_busy       db    0                         ;send task is busy
001D  00                C  recv_busy       db    0                         ;receive task is busy
001E  00                C  timer_int_busy  db    0                         ;timer interrupt is busy
001F  00                C  14_exit_busy    db    0                         ;14 exit routine is busy
0020  00                C  kb_int_busy     db    0                         ;keyboard interrupt is busy
0021  00                C  other_conn_busy db    0                         ;another (fs?) connection is busy
0022  00                C  windows_open    db    0                         ;windows are open
0023  00                C  do_open         db    0                         ;request to open windows
0024  00                C  do_stop         db    0                         ;request to stop all tasks
0025  00                C  first_dispatch  db    0                         ;request dispatch to initialize
0026  00                C  open_recvd      db    0                         ;14_openrcv returned TRUE
0027  00                C  send_msg_rqst   db    0                         ;send-msg request from kb_int
0028  00                C  suspended       db    0                         ;suspended by int16h special fct
0029  00                C  popped_up       db    0                         ;windows auto-popped up
```

```
                        level_four_interface

                        C
                        C
                        C  ;       Options
                        C
                        C
002A  00                C  test_mode       db      0 ;boolean              ;test mode is on
002B  00                C  auto_popup      db      0 ;boolean              ;windows popup automatically?
002C  00                C  timestamp       db      0 ;boolean              ;timestamp messages?
002D  00                C  auto_popdown    db      0 ;boolean              ;windows popdown automatically?
002E  010E              C  popdown_time    dw      15*18 ;(seconds*ticks)  ;auto popdown time
0030  00                C  channel         db      0                       ;channel (added to wks!)
                        C
0031  00                C  color           db      0                       ;color attribute modifier
0032  07                C  char_normal     db      07h                     ;normal character attribute
0033  0F                C  char_bold       db      0fh                     ;bold (highlighted) character
0034  01                C  char_line       db      01h                     ;underlined character
0035  09                C  char_boldline   db      09h                     ;bold underlined character
                        C
                        C
                        C  ;       Windows
                        C
                        C
                        C  ; (If you change these window sizes, check the window buffers as well!)
                        C  ; (You might also want to check the keyboard buffer size.)
                        C
0036  57 49 4E 44 4F 57 C                  db      'WINDOWS'               ;debug marker
      53                C
003D                    C  gwin_upperleft  label   word                    ;upper left corner of group window
003D  05                C  gwin_left_col   db      5
003E  00                C  gwin_top_row    db      0
003F                    C  gwin_lowerright label   word                    ;lower right corner of group window
003F  4E                C  gwin_right_col  db      78
0040  02                C  gwin_bottom_row db      2
                        C
0041                    C  swin_upperleft  label   word                    ;upper left corner of station window
0041  05                C  swin_left_col   db      5
0042  02                C  swin_top_row    db      2
0043                    C  swin_lowerright label   word                    ;lower right corner of station window
0043  4E                C  swin_right_col  db      78
0044  06                C  swin_bottom_row db      6
                        C
0045                    C  dwin_upperleft  label   word                    ;upper left corner of display window
0045  05                C  dwin_left_col   db      5
0046  06                C  dwin_top_row    db      6
0047                    C  dwin_lowerright label   word                    ;lower right corner of display window
0047  4E                C  dwin_right_col  db      78
0048  14                C  dwin_bottom_row db      20
                        C
0049                    C  twin_upperleft  label   word                    ;upper left corner of typing window
0049  05                C  twin_left_col   db      5
004A  14                C  twin_top_row    db      20
004B                    C  twin_lowerright label   word                    ;lower right corner of typing window
004B  4E                C  twin_right_col  db      78
004C  17                C  twin_bottom_row db      23
```

```
                                C
                                C
                                C  ;       Buffers
                                C
                                C
004D  42 55 46 46 45 52         C                  db      'BUFFERS'               ;debug marker
      53                        C
0054  00                        C  kbd_buffer_full db      0                       ;1 if buffer is full
0055  00 18                     C  kb_cursor       db      0,24                    ;keyboard cursor
                                C
0057  ????????                 C  send_msg_ptr    dd      ?                       ;ptr to kb_int send_msg buffer
005B  ??                        C  send_msg_stn    db      ?                       ;stn to send msg to
005C  ??                        C  send_msg_stat   db      ?                       ;status of kb_int send_msg
                                C
005D  14                        C  keychar         db      14h                     ;trigger is ctrl-t default
005E  00                        C  keychar_scan    db      0                       ;scan code if above is zero
                                C
                                C
= 008F                          C  kbd_bufsize     equ     2*(78-5-1)-1                    ;keyboard buffer size
= 0014                          C  from_to_size    equ     20                      ;max size of "from->to":
= 03C3                          C  rcv_bufsize     equ     kbd_bufsize+max_stn_list+max_grp_list+from_to_size
= 03C3                          C  snd_bufsize     equ     kbd_bufsize+max_stn_list+max_grp_list+from_to_size
                                C
005F     8F [                   C  kbd_buffer      db      kbd_bufsize dup(?)      ;keyboard buffer
              ??                C
                 ]              C
                                C
00EE     8F [                   C  kbd_buffer_prev db      kbd_bufsize dup(?)      ;previous keyboard buffer
              ??                C
                 ]              C
                                C
017D  03C3 [                    C  rcv_buffer      db      rcv_bufsize dup(?)      ;receive message buffer
              ??                C
                 ]              C
                                C
0540  03C3 [                    C  snd_buffer      db      snd_bufsize dup(?)      ;send message buffer
              ??                C
                 ]              C
                                C
                                C
0903  0000                      C  kbd_prevbufsiz  dw      0                       ;size of previous kbd line
0905  0000                      C  kbd_prevbufptr  dw      0                       ;offset into previous kbd line
0907  00                        C  kbd_prev_ins    db      0                       ;insert mode w/rt previous kbd line
                                C
0908  ????                      C  snd_buffer_size dw      ?                       ;size of message in snd_buffer
090A  ????                      C  snd_buffer_txt  dw      ?                       ;offset of start of text part
                                C
                                C
                                C
                                C  ;       Station and group lists
                                C
                                C
                                C  ; Note that various code walks through the stn_rec and grp_rec using lods
                                C  ; instructions which must be changed if fields are added or removed.
```

```
                        level_four_interface

                    C   ; Search for "stn_" or "grp_" to find those occurences.
                    C
                    C
090C  4C 49 53 54 53    C                   db      'LISTS'                 ;debug marker
                    C
= 0001                  C   ch_stn          equ     1                       ;stn list changed
= 0002                  C   ch_stn_atr      equ     2                       ;stn list attributes changed
= 0004                  C   ch_grp          equ     4                       ;grp list changed
= 0008                  C   ch_grp_atr      equ     8                       ;grp list attributes changed
                    C
0911  05                C   lists_chgd      db      ch_stn+ch_grp           ;station/group list changed flags
                    C
                    C
0912  0000              C   stn_list_size   dw      0                       ;current size of station list, +0
0914  01                C   stn_count       db      1                       ;count of stations
0915  00                C   rumor_count     db      0                       ;count of rumors
                    C
0916  0258 [          C   stn_list        db      max_stn_list dup(?)     ;station list, with variable-length
              ??
                  ]     C
                    C
                    C                                                   ; records as follows:
                    C   stn_rec         struc
0000  ??              C   stn_addr        db      ?                       ;station addr, or zero if end
0001  ??              C   stn_flags       db      ?                       ;station flags -- see below
0002  ????            C   stn_groupmask   dw      ?                       ;group membership mask
0004  ??              C   stn_namel       db      ?                       ;length of following name
0005  ?? ??           C   stn_name        db      ?,?;...                 ;station name
0007                C   stn_rec         ends
= 0005                  C   stn_rec_fsize   equ     5                       ;length of fixed part of stn_rec
                    C
                    C   ; The stn_groupmask is a set of bits representing group membership.  If the
                    C   ; first (msb) is on, we are a member of the first group in the group list,
                    C   ; and so on for up to 16 (max_groups) groups.
                    C
                    C   ; stn_flags bits:
                    C
= 0080                  C   rumor           equ     80h                     ;this stn is only a rumor, i.e.
                    C   ;                                               ;we got it from someone else
                    C
                    C
0B6E  0005              C   grp_list_size   dw      5                       ;current size of group list, +0
0B70  01                C   grp_count       db      1                       ;count of groups
                    C
0B71  03 41 4C 4C 00    C   grp_list        db      3,"ALL",0
0B76     C8 [           C                   db      max_grp_list dup (?)    ;group list, with records:
              ??
                  ]     C
                    C
                    C
                    C   grp_rec         struc
0000  ??              C   grp_namel       db      ?                       ;length of name
0001  ?? ??           C   grp_name        db      ?,?;...                 ;group name
0003                C   grp_rec         ends
```

```
                        level_four_interface

= 0001                  C  grp_rec_fsize   equ     1                       ;length of fixed part of grp_rec
                        C
0C3E     11 [           C  grp_number_map  db      max_groups+1 dup (0)    ;group number map[1..16]
            00          C
                 ]      C
                        C
                        C
                        C
                        C  ; Destination of messages.
                        C  ; Represents the relative group or station number (1..n) to which
                        C  ; messages should be sent.
                        C
0C4F  01                C  dest_grp        db      1                       ;relative group number
0C50  01                C  dest_stn        db      1                       ;relative station number
0C51  00                C  is_dest_stn     db      0                       ;1 if destination is stn
                        C
0C52  ??                C  send_ok_count   db      ?                       ;count of successful sends
0C53  ??                C  send_ng_count   db      ?                       ;count of unsuccessful sends
                        C
                        C
                        C  ;       Network stuff
                        C
                        C
0C54  4E 57             C                  db      'NW'                    ;debug marker
0C56  ??                C  rcv_arcnet      db      ?                       ;arcnet addr of stn rcvd from
0C57  ??                C  snd_arcnet      db      ?                       ;arcnet addr of stn sent to
0C58  00                C  brdcst_arcnet   db      0                       ;arcnet addr of stn which broadcast
0C59  ??                C  our_arcnet      db      ?                       ;our arcnet address
0C5A  ??                C  rcv_msg_type    db      ?                       ;incoming message type
0C5B  ??                C  last_open_rcv   db      ?                       ;last time we did open_recv
                        C                                                  ;  (low bits of clock only)
0C5C  0D04              C  wks             dw      talk_wks                ;our well-known socket
                        C                                                  ;  (modified by channel number)
0C5E  00 00 00 00       C  rcv_rb          dd      0                       ;ptrs to transport level
0C62  00 00 00 00       C  snd_rb          dd      0                       ;   request blocks (rbs)
0C66  ????????          C  l4_publics      dd      ?                       ;ptr to L4 publics
0C6A  0000              C  our_l4_debug    dw      0                       ;the l4 debug value
0C6C  ????????          C  old_l4_exit     dd      ?                       ;existing l4 exit rtn
                        C
                        C
                        C
                        C  ;       DOS stuff
                        C
                        C
0C70  44 4F 53          C                  db      'DOS'                   ;debug marker
0C73  ????              C  psp             dw      ?                       ;ptr to dos psp
                        C
0C75  ????????          C  old_key_vector  dd      ?                       ;existing key interrupt rtn
0C79  ????????          C  old_time_vec    dd      ?                       ;existing timer interrupt rtn
                        C
0C7D  0000              C  ticks           dw      0                       ;count of timer ticks
                        C
0C7F  0000              C  tweedle_ptr     dw      0                       ;offset into tweedle array
                        C
```

```
                                level_four_interface

0C81  08 0A 08 00           C  tweedle1        db      8,10,8,0                ;ROLM-like phone for background msg
0C85  0E 0C 0B 00           C  tweedle2        db      14,12,11,0              ;low-med-high for window popup
0C89  09 00                 C  tweedle3        db      9,0                     ;short beep for incoming msg
0C8B  04 03 02 00           C  tweedle4        db      4,3,2,0                 ;chirp for new user
0C8F  3C 3C 00              C  tweedle5        db      60,60,0                 ;low boop for failure to send
0C92  0F 0F 0F 0B 0B 0B     C  tweedle6        db      15,15,15,11,11,11       ;ambulance sound for errors
0C98  0F 0F 0F 0B 0B 0B     C                  db      15,15,15,11,11,11,0
      00                    C
                            C
0C9F  0A                    C  kb10            db      10
0CA0  0444                  C  kw1092          dw      1092                    ;18.2 * 60 for time conversion
                            C
                            C
                            C  ;       Task stacks
                            C  ;
                            C  ; Note that ALL stacks must be large, because the keyboard interrupt process
                            C  ; can be recursive without bound when the keyboard buffer overflows.  This is
                            C  ; a serious and fatal flaw in the rom bios.  Try it with virgin ibm software!
                            C  ; You too can crash your machine by typing fast!
                            C  ;
                            C
0CA2  53 54 41 43 4B 53     C                  db      'STACKS'                ;debug marker
0CA8  0258 [                C                  db      600 dup(70h)            ;0: keyboard interrupt process
              70            C
                   ]        C
                            C
0F00                        C  stack0          label   word
0F00  0258 [                C                  db      600 dup(71h)            ;1: keyboard process
              71            C
                   ]        C
                            C
1158                        C  stack1          label   word
1158  0258 [                C                  db      600 dup(72h)            ;2: receive process
              72            C
                   ]        C
                            C
13B0                        C  stack2          label   word
13B0  0258 [                C                  db      600 dup(73h)            ;3: send process
              73            C
                   ]        C
                            C
1608                        C  stack3          label   word
                            C
                            C
                            C  ;
                            C  ; Window buffers.  At some point we could move them to between the resident
                            C  ; code and the initialization code, so they can be shrunk to fit.
                            C  ; At the moment they are exactly the size needed and are part of the resident
                            C  ; segment.
                            C  ;
                            C  ; Note that the size of the user window can be changed by an invocation
                            C  ; parameter, and the size of the display window changes to accomodate it.
                            C  ; Thus the boundaries between the various windows may not be exactly as
                            C  ; shown below, but the total space will be no more than what we allocate.
```

```
                        level_four_interface

                    C   ;
                    C   ; Remember to change the window definitions up above if you change window
                    C   ; sizes, and also check the initialization code.
                    C   ;
                    C
1608  53 43 52 45 45 4E C                   db      'SCREENS'               ;debug marker
      53            C
                    C
160F                C   window_buffers  label   byte                    ;start of all the window buffers
160F                C   group_window    label   byte                    ;start of the group window
                    C
                    C   ;               #pages      height      width    2 bytes per char
                    C   ;               -------     --------    --------  --
160F  01BC [        C           db          1     * (2-0+1)   * (78-5+1) * 2 dup (?)  ;the group window
            ??      C
                ]   C
                    C
17CB  01BC [        C           db          1     * (4-2+1)   * (78-5+1) * 2 dup (?)  ;the station window
            ??      C
                ]   C
                    C
1987  1D7C [        C           db      n_pages   * (20-4+1)  * (78-5+1) * 2 dup (?)  ;the display window
            ??      C
                ]   C
                    C
3703  0250 [        C           db          1     * (23-20+1)* (78-5+1) * 2 dup (?)  ;the typing window
            ??      C
                ]   C
                    C
                    C
                    C
3953                C   vseg            ends
                    C

0000                    cseg            segment public 'code'
                                        assume  cs:cgrp,es:nothing,ds:nothing

                                        public  send_task,rcv_task,kb_task
                                        public  key_int_rtn,timer_int_rtn
                                        public  to_hex,release_rbs,talk_exit
                                        public  l4_entries,al4locate,l4_exit_rtn

                        ;-----------------------------------------------------------------------------
                        ;
                        ;       Level 4 Initialization Routines.
                        ;
                        ;-----------------------------------------------------------------------------

                                        subttl

                        ;;              include e:l4loc.asm
                                        .list
```

```
                        page

                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                        ;
                        ;  Keyboard task
                        ;
                        ;       Open and close windows as required.
                        ;       Read a line from the keyboard, displaying it in the typing window.
                        ;       When done, queue it to be transmitted to all stations, and then
                        ;       move it to the display window.  Repeat.
                        ;
                        ;       If the station list has changed, redisplay the station window.
                        ;
                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                                %out    -----kb_task

00B5                    kb_task         proc    near

                         $do                    ;-------- keyboard task infinite loop --------
                                                 *

                             $do                ;until we can shut down the tasks

                        ;       Open windows if requested to do so and no other rb's are active
                        ;       and we are not in the timer or 14 interrupt routines

00B5  2E: F6 06 0023 R 01               test    do_open,1               ;request to open windows?
                                        $ifnot z                        ;yes
00BD  2E: A0 001F R                     mov     al,14_exit_busy         ;in 14 exit?
00C1  2E: 0A 06 001E R                  or      al,timer_int_busy       ;or timer int?
00C6  2E: 0A 06 0021 R                  or      al,other_conn_busy      ;or open connection?
                                        $if     z                       ;no: ok
00CD  2E: C6 06 0023 R 00                 mov   do_open,0               ;clear window open request
00D3  E8 00FC R                           call  process_kb              ;process keyboard commands
                                        $endif
                                       $endif

                        ;       Process messages in the background until everything is quiet

00D6  E8 0E69 R                         call    dispatch                ;run other tasks
00D9  2E: A0 001D R                     mov     al,recv_busy            ;until nothing being received
00DD  2E: 0A 06 0026 R                  or      al,open_recvd           ;or about to be
00E2  2E: 0A 06 001C R                  or      al,send_busy            ;or going out

                             $repeatuntil z


                        ;       Shut down the tasks

00E9  2E: C6 06 0024 R 01               mov     do_stop,1               ;flag to stop tasks
```

```
                                        $do
00EF  E8 0E69 R                           call  dispatch
00F2  2E: F6 06 0024 R 01                 test  do_stop,1             ;and wait until it happened
                                        $repeatuntil z

                        |       $repeat              ;-------- end infinite loop ----------

00FC                            kb_task         endp
```

```
                                page
                        ;-------------------------------------------------------------------------
                        ;
                        ;   Process keyboard:   Open the windows, read lines until "Q", then
                        ;                       close the windows.
                        ;
                        ;-------------------------------------------------------------------------

00FC                    process_kb      proc    near
00FC  8C C8                             mov     ax,cs
00FE  8E D8                             mov     ds,ax
0100  8E C0                             mov     es,ax
                                        assume  ds:cgrp,es:cgrp

                        ;       Open the windows

0102  B8 0604                           mov     ax,select_gwin
0105  E8 0000 E                         call    window                  ;open group window
0108  B4 01                             mov     ah,window_open
010A  E8 0000 E                         call    window
                                        $if     c                       ;if it failed (graphics mode)
010F  8D 06 0C92 R                        lea   ax,tweedle6
0113  A3 0C7F R                           mov   tweedle_ptr,ax          ;error sound
0116  E9 0387 R                           jmp   process_kb_ret          ;and return
                                        $endif
0119  B8 0603                           mov     ax,select_swin
011C  E8 0000 E                         call    window                  ;open station window
011F  B4 01                             mov     ah,window_open
0121  E8 0000 E                         call    window
0124  B8 0601                           mov     ax,select_dwin
0127  E8 0000 E                         call    window                  ;open display window
012A  B4 01                             mov     ah,window_open
012C  E8 0000 E                         call    window
012F  B0 04                             mov     al,4                    ;do an "END" page operation
0131  B4 09                             mov     ah,window_pageop
0133  E8 0000 E                         call    window
0136  B8 0602                           mov     ax,select_twin
0139  E8 0000 E                         call    window                  ;open typing window
013C  B4 01                             mov     ah,window_open
013E  E8 0000 E                         call    window
0141  C6 06 0022 R 01                   mov     windows_open,1          ;flag: windows open


                         $do    ;--------------------                   ;repeat until Q

                        ;       Read a line from the keyboard

0146  B0 3A                             mov     al,":"                  ;print ":"
0148  E8 0D92 R                         call    w_print
014B  BE 0000           zero_line:      mov     si,0                    ;buffer index
014E  89 36 0905 R                      mov     kbd_prevbufptr,si       ;previous buffer pointer
0152  C6 06 0907 R 00                   mov     kbd_prev_ins,0          ;not insert mode
```

```
                                ;               Wait for a key

0157  C7 06 0C7D R 0000                         mov     ticks,0                 ;reset the tick counter

015D                            get_kb_char:    $do
015D  E8 0E69 R                                   call  dispatch                ;dispatch tasks
0160  80 3E 0911 R 00                             cmp   lists_chgd,0            ;has the stationlist changed?
                                                  $ifnot e
0167  E8 0397 R                                    call show_stations           ;yes: display the new list
016A  C6 06 0911 R 00                              mov  lists_chgd,0            ;  and reset the flag
                                                  $endif

016F  F6 06 0029 R 01                             test  popped_up,1             ;did windows popup automatically?
                                                  $ifnot z
0176  F6 06 002D R 01                             test  auto_popdown,1          ;does he want auto-popdown?
                                                  $ifnot z
017D  A1 0C7D R                                     mov ax,ticks                ;yes: check if time expired
0180  3B 06 002E R                                  cmp ax,popdown_time
                                                    $if ae
0186  46                                             inc si                     ;(so : is erased)
0187  E9 034B R                                      jmp do_close               ;do auto-popdown close of windows
                                                    $endif
                                                  $endif
                                                  $endif

018A  B4 01                                       mov   ah,key_stat
018C  CD 16                                       int   key_int               ;any key pressed?
                                                $repeatwhile z                  ;no... keep trying

0190  B4 00                                     mov     ah,key_read
0192  CD 16                                     int     key_int                 ;read the key

0194  C6 06 0029 R 00                           mov     popped_up,0             ;cancel auto-popdown

                                ;       Process special characters

0199  3C 08                                     cmp     al,bs                   ;************  backspace?
                                                $if     e
019D  83 FE 00                                    cmp   si,0
01A0  74 BB                                       je    get_kb_char             ;ignore it if at start of input buffer
01A2  E8 0D92 R                                   call  w_print                 ;otherwise print bs
01A5  4E                                          dec   si                      ;and decrement index
                                                  $if   c                       ;word wrap backspace:
                                                    $do                         ;  move to after 1st nonblank
01A8  80 BC 005F R 20                                 cmp kbd_buffer[si],'
                                                      $exitif ne
01AF  4E                                              dec si
                                                    $repeatuntil s
01B2  46                                            inc si
                                                  $endif                        ;word wrap
01B3  EB A8                                       jmp   short get_kb_char
```

```
                                        $endif                          ;backspace


01B5  3C 0A                             cmp     al,lf                   ;*************  ignore linefeeds
01B7  74 A4                             je      get_kb_char

01B9  3C 07                             cmp     al,bell                 ;*************  ignore bells
01BB  74 A0                             je      get_kb_char


01BD  3C 1B                             cmp     al,esc                  ;**************   escape?
                                        $if     e
01C1  8B CE                               mov   cx,si
01C3  B0 08                               mov   al,bs                   ;print bs until start of buffer
                                          $do   cxnz
01C7  E8 0D92 R                            call w_print
                                          $repeatloop
01CC  E9 014B R                           jmp   zero_line
                                        $endif


01CF  3A 06 005D R                      cmp     al,keychar              ;***************  trigger char
01D3  75 0E                               jne   not_trig
01D5  0A C0                             or      al,al
                                        $if     z
01D9  3A 26 005E R                        cmp   ah,keychar_scan         ;(also check extended scancode)
01DD  75 04                               jne   not_trig
                                        $endif
01DF  46                                inc     si                      ;(so that ":" is erased)
01E0  E9 034B R                         jmp     do_close
01E3                            not_trig:

01E3  3D 4800                           cmp     ax,key_up               ;****************  up-arrow key:
                                        $if     e                       ;           group destination
01E8  C6 06 0C51 R 00                     mov   is_dest_stn,0           ;not stn destination
01ED  80 0E 0911 R 0A                     or    lists_chgd,ch_grp_atr+ch_stn_atr
01F2  E9 015D R                           jmp   get_kb_char
                                        $endif


01F5  3D 5000                           cmp     ax,key_down             ;*****************  down-arrow key:
                                        $if     e                       ;            stn destination
01FA  C6 06 0C51 R 01                     mov   is_dest_stn,1           ;yes stn destination
01FF  80 0E 0911 R 0A                     or    lists_chgd,ch_grp_atr+ch_stn_atr
0204  E9 015D R                           jmp   get_kb_char
                                        $endif


0207  3D 4D00                           cmp     ax,key_right            ;******************  right-arrow key:
                                        $if     e                       ;          next destination
020C  F6 06 0C51 R 01                     test  is_dest_stn,1
                                          $ifnot z                      ;next station
                                  if not show_rumors
                                            $do
```

```
                                        endif
0213  A0 0C50 R                                mov al,dest_stn
0216  FE C0                                    inc al
0218  3A 06 0914 R                             cmp al,stn_count
                                               $if a
021E  B0 01                                     mov al,1
                                               $endif
0220  A2 0C50 R                                mov dest_stn,al
                                        if not show_rumors
                                               call is_dest_rumor          ;is it a rumor?
                                              $repeatuntil z               ;yes - try next
                                        endif
0223  80 0E 0911 R 02                          or   lists_chgd,ch_stn_atr
                                             $else                         ;next group
022B  A0 0C4F R                              mov  al,dest_grp
022E  FE C0                                  inc  al
0230  3A 06 0B70 R                          cmp  al,grp_count
                                             $if  a
0236  B0 01                                   mov al,1
                                             $endif
0238  A2 0C4F R                            mov   dest_grp,al
023B  80 0E 0911 R 0A                      or    lists_chgd,ch_grp_atr+ch_stn_atr
                                           $endif
0240  E9 015D R                           jmp     get_kb_char
                                          $endif


0243  3D 4B00                             cmp     ax,key_left               ;*****************  left-arrow key:
                                           $if     e                         ;       previous destination
0248  F6 06 0C51 R 01                        test  is_dest_stn,1
                                             $ifnot z                        ;previous station
                                        if not show_rumors
                                             $do
                                        endif
024F  A0 0C50 R                                mov al,dest_stn
0252  FE C8                                    dec al
                                               $if z
0256  A0 0914 R                                 mov al,stn_count
                                               $endif
0259  A2 0C50 R                                mov dest_stn,al
                                        if not show_rumors
                                               call is_dest_rumor          ;is it a rumor?
                                              $repeatuntil z               ;yes: try previous
                                        endif
025C  80 0E 0911 R 02                          or   lists_chgd,ch_stn_atr
                                             $else                         ;previous group
0264  A0 0C4F R                              mov  al,dest_grp
0267  FE C8                                  dec  al
                                             $if  z
026B  A0 0B70 R                               mov al,grp_count
                                             $endif
026E  A2 0C4F R                              mov  dest_grp,al
0271  80 0E 0911 R 0A                        or   lists_chgd,ch_grp_atr+ch_stn_atr
                                           $endif
```

```
0276  E9 015D R                        jmp     get_kb_char
                                       $endif


0279  3D 3B00                          cmp     ax,key_f1               ;*****************  F1 key:
                                       $if     e                       ;        copy previous buffer char
027E  3B 36 0903 R                       cmp   si,kbd_prevbufsiz
                                         $if   l
0284  8A 84 00EE R                         mov al,kbd_buffer_prev[si]  ;use it if it exists
0288  32 E4                                xor ah,ah
                                         $else
028D  E9 015D R                           jmp  get_kb_char             ;otherwise ignore
                                         $endif
                                       $endif


0290  3D 3D00                          cmp     ax,key_f3               ;*****************  F3 key:
                                       $if     e                       ;        copy previous buffer
0295  8B 0E 0903 R                      mov   cx,kbd_prevbufsiz        ;amount in previous buffer
0299  2B CE                             sub   cx,si                    ;less amount typed in this buffer
                                         $if   g                       ;if something left
029D  8D BC 005F R                        lea  di,kbd_buffer[si]       ;copy to end of current buffer
02A1  8D B4 00EE R                        lea  si,kbd_buffer_prev[si]  ;move tail of previous buffer
                                          $do
02A5  AC                                   lods kbd_buffer             ;move characters
02A6  AA                                   stos kbd_buffer_prev
02A7  E8 0D92 R                            call w_print                ;and print them
                                          $repeatloop
02AC  8B 36 0903 R                        mov  si,kbd_prevbufsiz       ;# of characters now in buffer
                                         $endif
02B0  E9 015D R                        jmp     get_kb_char
                                       $endif


02B3  3D 4700                          cmp     ax,key_home             ;HOME key
                                       $if     e
02B8  B3 03                              mov   bl,3
02BA  B8 0601                do_pageop:  mov   ax,select_dwin           ;select display window
02BD  E8 0000 E                          call  window
02C0  8A C3                              mov   al,bl                   ;pageop code
02C2  B4 09                              mov   ah,window_pageop
02C4  E8 0000 E                          call  window
02C7  B8 0602                            mov   ax,select_twin          ;reselect typing window
02CA  E8 0000 E                          call  window
02CD  E9 015D R                          jmp   get_kb_char
                                       $endif

02D0  3D 4F00                          cmp     ax,key_end              ;**************  END key
                                       $if     e
02D5  B3 04                              mov   bl,4
02D7  EB E1                              jmp   short do_pageop
                                       $endif

02D9  3D 4900                          cmp     ax,key_pgup             ;**************  PGUP key
                                       $if     e
```

```
02DE  B3 01                                   mov   bl,1
02E0  EB D8                                   jmp   short do_pageop
                                            $endif

02E2  3D 5100                               cmp     ax,key_pgdn             ;***************  PGDN key
                                            $if     e
02E7  B3 02                                   mov   bl,2
02E9  EB CF                                   jmp   short do_pageop
                                            $endif

02EB  0A C0                                 or      al,al                   ;Ignore all other extended codes
                                            $if     z
02EF  E9 015D R                               jmp   get_kb_char
                                            $endif


                            ;       Store and display the character


02F2  88 84 005F R                          mov     kbd_buffer[si],al       ;store char in buffer
02F6  46                                    inc     si                      ;update input buffer index

02F7  3C 0D                                 cmp     al,cr                   ;carriage return ends the line
02F9  74 10                                 je      got_line
02FB  81 FE 008F                            cmp     si,kbd_bufsize          ;check for buffer overflow
                                            $if     l
0301  E8 0D92 R                               call  w_print                 ;not: display it
                                            $else
0307  4E                                      dec   si                      ;else pin at end-of-buffer
                                            $endif
0308  E9 015D R                             jmp     get_kb_char


                            ;       Check for "Q" or "q" and return if so

030B  83 FE 02              got_line:       cmp     si,2
030E  75 0E                                 jne     not_quit
0310  80 3E 005F R 51                       cmp     kbd_buffer,"Q"
                                            $exitif e
0317  80 3E 005F R 71                       cmp     kbd_buffer,"q"
                                            $exitif e
031E                        not_quit:


                            ;       Process the line


031E  B0 0D                                 mov     al,cr                   ;echo cr
0320  E8 0D92 R                             call    w_print
0323  83 FE 01                              cmp     si,1                    ;is line null?  (just cr)
                                            $ifnot  e
0328  C6 06 0054 R 01                       mov     kbd_buffer_full,1       ;no: signal buffer full to send_task
032D  8B CE                                 mov     cx,si                   ;copy to "previous" buffer
032F  49                                    dec     cx                      ; (without CR)
```

```
0330  89 0E 0903 R                        mov     kbd_prevbufsiz,cx
0334  8D 36 005F R                        lea     si,kbd_buffer
0338  8D 3E 00EE R                        lea     di,kbd_buffer_prev
033C  F3/ A4                              rep     movsb
                                          $do
033E  E8 0E69 R                             call  dispatch
0341  F6 06 0054 R 01                      test  kbd_buffer_full,1       ;and wait until kbd_buffer is free
                                          $repeatuntil z
                                          $endif                        ;null check

                           $repeat        ;--------------------          ;repeat until Q

034B                       do_close:
034B  B0 08                               mov     al,bs                 ;simulate backspace over
034D  8B CE                               mov     cx,si                 ; anything in the buffer
                                          $do     cxnz
0351  E8 0D92 R                             call  w_print               ;  example: ":q"
                                          $repeatloop


                           ;       Close the windows and return

0356  B8 0602                             mov     ax,select_twin
0359  E8 0000 E                          call    window                ;close typing window
035C  B4 02                               mov     ah,window_close
035E  E8 0000 E                          call    window
0361  B8 0601                             mov     ax,select_dwin
0364  E8 0000 E                          call    window                ;close display window
0367  B4 02                               mov     ah,window_close
0369  E8 0000 E                          call    window
036C  B8 0603                             mov     ax,select_swin
036F  E8 0000 E                          call    window                ;close station window
0372  B4 02                               mov     ah,window_close
0374  E8 0000 E                          call    window
0377  B8 0604                             mov     ax,select_gwin
037A  E8 0000 E                          call    window                ;close group window
037D  B4 02                               mov     ah,window_close
037F  E8 0000 E                          call    window
0382  C6 06 0022 R 00                     mov     windows_open,0        ;flag: windows close

0387                       process_kb_ret:
0387  C3                                 ret

0388                       process_kb     endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ; show_stations        Display known or rumored stations in the station-window,
                        ;                      and groups in the group window.  Highlight the current
                        ;                      destination station or group (and station members).
                        ;                      Underline the station being sent to, if any.
                        ;
                        ;                      Use lists_chgd bits to know what has changed.  Try to
                        ;                      minimize window glitching by overwriting instead of
                        ;                      clearing and rewriting the windows when possible.
                        ;
                        ;                      Destroys ax, bx, cx, dx.
                        ;
                        ;-----------------------------------------------------------------------

0388  47 72 6F 75 70 73 show_stn_grpmsg db      'Groups: '
      3A 20
= 0008                  show_stn_grpmsgl equ    $-show_stn_grpmsg

0390  55 73 65 72 73 3A show_stn_usrmsg db      'Users: '
      20
= 0007                  show_stn_usrmsgl equ    $-show_stn_usrmsg

0397                    show_stations   proc
0397  56                                push    si

0398  F6 06 0911 R 0C           test    lists_chgd,ch_grp+ch_grp_atr    ;any group changes?
                           $if     z
039F  E9 042D R              jmp     show_stn_stns
                           $endif


                        ;       Do group window

03A2  B8 0604                       mov     ax,select_gwin          ;select the group window
03A5  E8 0000 E                     call    window
03A8  F6 06 0911 R 04               test    lists_chgd,ch_grp       ;must be complete rewrite?
                                    $ifnot  z
03AF  B0 0D                           mov   al,cr                   ;yes: clear it with CR
03B1  E8 0D92 R                       call  w_print
                                    $endif
03B4  BA 0101                       mov     dx,0101h                ;cursor to start
03B7  B4 05                         mov     ah,window_setcurs
03B9  E8 0000 E                     call    window
03BC  F6 06 0C51 R 01               test    is_dest_stn,1           ;if group destination,
                                    $if     z
03C3  B4 08                           mov   ah,window_setattr
03C5  A0 0033 R                       mov   al,char_bold            ;set bold
03C8  E8 0000 E                       call  window
                                    $endif
03CB  8D 36 0388 R                  lea     si,show_stn_grpmsg      ;print initial message
03CF  B9 0008                       mov     cx,show_stn_grpmsgl
```

```
                                        $do
03D2  AC                                  lodsb
03D3  E8 0D92 R                           call  w_print                 ;"Groups: "
                                        $repeatloop
03D8  B4 08                             mov     ah,window_setattr
03DA  A0 0032 R                         mov     al,char_normal
03DD  E8 0000 E                         call    window

03E0  8D 36 0B71 R                      lea     si,grp_list
03E4  32 DB                             xor     bl,bl                 ;group counter
                                        $do
03E6  FE C3                               inc   bl                    ;count next group
03E8  AC                                  lods  grp_list.grp_namel    ;length of name
03E9  0A C0                               or    al,al
                                          $exitif z                   ;done if zero
                                        if      assert
03ED  3C 10                               cmp   al,max_namesize
                                          $if   g                     ;internal error!
03F1  8B DE                                 mov bx,si
03F3  8A C8                                 mov cl,al
03F5  32 ED                                 xor ch,ch
03F7  B0 02                                 mov al,2
03F9  E8 0E0A R                             call int_error
03FC  EB 2F 90                              jmp show_stn_stns
                                          $endif
                                        endif
03FF  32 ED                               xor   ch,ch
0401  8A C8                               mov   cl,al                 ;length of name to cx
0403  F6 06 0C51 R 01                     test  is_dest_stn,1         ;group destination currently?
                                          $if   z
040A  3A 1E 0C4F R                         cmp  bl,dest_grp           ;this group?
                                           $if  e
0410  B4 08                                 mov ah,window_setattr
0412  A0 0033 R                             mov al,char_bold          ;yes: set bold
0415  E8 0000 E                             call window
                                           $endif
                                          $endif
                                          $do
0418  AC                                   lods grp_list.grp_name     ;print it
0419  E8 0D92 R                            call w_print
                                          $repeatloop
041E  B4 08                               mov   ah,window_setattr
0420  A0 0032 R                           mov   al,char_normal
0423  E8 0000 E                           call  window
0426  B0 20                               mov   al," "                ;blank as separator
0428  E8 0D92 R                           call  w_print
                                        $repeat                       ;next group


                              ;       Do station window

042D  B8 0603                 show_stn_stns:  mov     ax,select_swin  ;select the station window
0430  E8 0000 E                               call    window
0433  F6 06 0911 R 01                         test    lists_chgd,ch_stn ;must be complete rewrite?
```

```
                                        $ifnot  z
043A  B0 0D                               mov   al,cr                   ;yes: clear the window with CRs
043C  8A 0E 0044 R                        mov   cl,swin_bottom_row      ; number of CRs needed
0440  2A 0E 0042 R                        sub   cl,swin_top_row         ; is 2*(bottom-top-1)-1,
0444  32 ED                              xor   ch,ch                   ; or 2*(bottom-top)-3.
0446  D1 E1                              sal   cx,1                    ; (Derivation left, naturally,
0448  83 E9 03                           sub   cx,3                    ;  as an exercise for the reader.)
                                          $do
044B  E8 0D92 R                            call w_print
                                          $repeatloop
                                        $endif
0450  BA 0101                           mov     dx,0101h              ;cursor to start
0453  B4 05                             mov     ah,window_setcurs
0455  E8 0000 E                        call    window

0458  F6 06 0C51 R 01                   test    is_dest_stn,1         ;if single user destination,
                                        $ifnot  z
045F  B4 08                               mov   ah,window_setattr
0461  A0 0033 R                          mov   al,char_bold          ;set bold
0464  E8 0000 E                          call window
                                        $endif
0467  8D 36 0390 R                      lea     si,show_stn_usrmsg    ;print initial message
046B  B9 0007                           mov     cx,show_stn_usrmsgl
                                        $do
046E  AC                                  lodsb
046F  E8 0D92 R                           call  w_print               ;"Users: "
                                        $repeatloop
0474  B4 08                             mov     ah,window_setattr
0476  A0 0032 R                         mov     al,char_normal
0479  E8 0000 E                        call    window

047C  8D 36 0916 R                      lea     si,stn_list           ;pointer into station table
0480  32 DB                            xor     bl,bl                 ;station counter
                                    $do                             ;loop for all station recs
0482  FE C3                             inc     bl                    ;count next station

0484  AC                                lods    stn_list.stn_addr     ;station address
0485  0A C0                            or      al,al
                                        $exitif z                     ;end of list if zero
0489  8A F0                            mov     dh,al                 ;dh=station address

048B  AC                                lods    stn_list.stn_flags    ;station flags
048C  8A D0                             mov     dl,al                 ;save in dl
                                        if      not show_rumors
                                          test  dl,rumor              ;is it a rumor?
                                          jnz   show_stn_next         ;yes: don't show it
                                        endif

048E  AD                                lods    stn_list.stn_groupmask ;get groupmask in ax
048F  8A 16 0032 R                      mov     dl,char_normal        ;default character attribute
0493  3A 36 0C57 R                      cmp     dh,snd_arcnet         ;are we sending to this station
                                        $if     e                     ; right now?
0499  8A 16 0034 R                        mov   dl,char_line          ; yes: underline it
                                        $endif
```

```
049D  F6 06 0C51 R 01                    test    is_dest_stn,1           ;is destination a stn?
                                         $ifnot  z
04A4  3A 1E 0C50 R                         cmp   bl,dest_stn             ;yes. this stn?
04A8  74 0C                                je    show_stn_bold           ;yes: bold it
                                         $else
04AD  8A 0E 0C4F R                         mov   cl,dest_grp             ;number of destination group
04B1  F8                                   clc
04B2  D3 D0                                rcl   ax,cl                   ;look at that groupmask bit
                                           $if   c                       ;if on, bold it
04B6  8A 16 0033 R           show_stn_bold:  mov  dl,char_bold           ;set bold
04BA  3A 36 0C57 R                           cmp  dh,snd_arcnet          ;are we sending to this station
                                             $if  e                      ; right now?
04C0  8A 16 0035 R                            mov dl,char_boldline; yes: bold underline
                                             $endif
                                           $endif
                                         $endif
04C4  B4 08                              mov     ah,window_setattr       ;set normal, bold, underline,
04C6  8A C2                              mov     al,dl                   ; or bold underline
04C8  E8 0000 E                          call    window

04CB  AC                                 lods    stn_list.stn_namel      ;length of name
                                         if      assert
04CC  3C 10                                cmp   al,max_namesize
                                           $if   g                       ;internal error!
04D0  8B DE                                  mov bx,si
04D2  8A C8                                  mov cl,al
04D4  32 ED                                  xor ch,ch
04D6  B0 03                                  mov al,3
04D8  E8 0E0A R                              call int_error
04DB  EB 23 90                               jmp show_stn_z
                                           $endif
                                         endif
04DE  98                                 cbw
04DF  8B C8                              mov     cx,ax
                                         $do
04E1  AC                                   lods  stn_list.stn_name
04E2  E8 0D92 R                            call  w_print                 ,print the name
                                         $repeatloop
04E7  F6 C2 80                           test    dl,rumor                ;is this stn only a rumor?
                                         $ifnot  z
04EC  B0 3F                                mov   al,"?"                  ;yes: flag it
04EE  E8 0D92 R                            call  w_print
                                         $endif
04F1  B4 08                              mov     ah,window_setattr
04F3  A0 0032 R                          mov     al,char_normal          ;unbold
04F6  E8 0000 E                          call    window
04F9  B0 20                              mov     al," "
04FB  E8 0D92 R                          call    w_print                 ;blank as separator
04FE                         show_stn_next:
                                       $repeat

0500  B8 0602                show_stn_z: mov     ax,select_twin          ;return to the typing window
0503  E8 0000 E                         call    window
0506  5E                                pop     si
```

```
0507  C3                                ret
0508                          show_stations   endp


                              ;
                              ; is_dest_rumor         Is the dest_stn a rumor?  Return NZ if so.
                              ;
                              ;       Destroys: cx
                              ;

                                              if      not show_rumors
                              is_dest_rumor   proc    near
                                              push    si
                                              lea     si,stn_list             ;start of station list
                                              xor     ch,ch
                                              mov     cl,dest_stn
                                              dec     cl                      ;dest_stn is origin 1
                                              $do     cxnz
                                                mov   bl,[si].stn_namel       ;skip to next station
                                              ;assert bl<>0
                                                xor   bh,bh
                                                lea   si,stn_rec_fsize[si+bx]
                                              $repeatloop                     ;count stations
                                              test    [si].stn_flags,rumor    ;set NZ if rumor
                                              pop     si
                                              ret
                              is_dest_rumor   endp
                                              endif

                                              assume  ds:nothing,es:nothing
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                        ;
                        ; Network receive task
                        ;                               .
                        ;       Receive a station-list and text message.
                        ;       Add the stations to our list, and display the message.
                        ;
                        ;       If a broadcast is received, signal the send task to send a
                        ;       station-list
                        ;
                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                                %out    -----rcv_task

0508                    rcv_task        proc    near
0508  2E: C5 36 0C5E R                 lds     si,rcv_rb               ;get ptr to receive rb in ds:si
050D  8C C8                           mov     ax,cs
050F  8E D8                           mov     ds,ax                   ;ds:=cs
0511  8E C0                           mov     es,ax                   ;es:=cs
                                      assume  ds:cgrp,es:cgrp



                                $do     ;**********************;infinite loop of receive task

0513  E8 0767 R                       call    recv_message            ;get a message

0516  80 3E 0C5A R 99                 cmp     rcv_msg_type,genl_msg_type
                                      $if     e                       ;---regular text message
051D  E8 055E R                         call  process_msg             ;process stn list and text
                                      $endif

0520  80 3E 0C5A R 98                 cmp     rcv_msg_type,logon_msg_type
                                      $if     e                       ;---logon broadcast message
0527  E8 055E R                         call  process_msg             ;process the stationlist part
                                        $do                           ;now send our stationlist to him
052A  E8 0E69 R                          call dispatch
052D  80 3E 0C58 R 00                    cmp  brdcst_arcnet,0          ;wait if any prior send is queued
                                        $repeatuntil e
0534  A0 0C56 R                         mov   al,rcv_arcnet
0537  A2 0C58 R                         mov   brdcst_arcnet,al;queue this one
                                      $endif

053A  80 3E 0C5A R 97                 cmp     rcv_msg_type,logoff_msg_type
                                      $if     e                       ;---logoff message
0541  8A 1E 0C56 R                     mov   bl,rcv_arcnet
0545  E8 0C68 R                        call  stn_lookup              ;lookup it's name (stn=bl)
                                        $ifnot z                      ;if found,
054A  E8 0CC0 R                          call stn_remove              ;delete it
                                ;;bug: if dest_stn is >= removed station, it was changed!
054D  80 0E 0911 R 01                    or   lists_chgd,ch_stn       ;bug coverup;  should fix this!
```

```
0552  C6 06 0C50 R 01                        mov  dest_stn,1
                                            $endif
                                          $endif

0557  C6 06 001D R 00                      mov     recv_busy,0          ;clear busy flag

                                 $repeat ;*************************      ;end infinite loop of receive task

055E                          rcv_task     endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ; process_msg   Process the incoming message.
                        ;
                        ;               Messages are a sequence of message subtypes, ending
                        ;               with a message subtype of zero.
                        ;
                        ;-----------------------------------------------------------------------

055E                    process_msg     proc
055E  8D 3E 017D R                      lea     di,rcv_buffer           ;di points into buffer always

                                $do                                     ;process all subtypes

0562  8A 05                             mov     al,[di]                 ;get subtype
0564  47                                inc     di
0565  0A C0                             or      al,al
                                        $exitif z                       ;stop if zero

0569  3C BB                             cmp     al,grp_subtype          ;process according to subtype
                                        $if     e
056D  E8 0599 R                           call  process_grp_msg
                                        $else
0573  3C AA                             cmp     al,stn_subtype
                                        $if     e
0577  E8 060E R                           call  process_stn_msg
                                        $else
057D  3C CC                             cmp     al,txt_subtype
                                        $if     e
0581  E8 06FD R                           call  process_txt_msg
                                        $else                           ;bad subtype
                                          if    assert
0587  8A 3E 0C56 R                         mov   bh,rcv_arcnet
058B  8A D8                               mov   bl,al
058D  8B CF                               mov   cx,di
058F  B0 01                               mov   al,1                    ;internal error code 1
0591  E8 0E0A R                           call  int_error
0594  EB 02                               jmp   short process_msg_z
                                          endif
                                        $endif
                                        $endif
                                        $endif

                                $repeat

0598  C3                       process_msg_z:  ret
0599                           process_msg     endp
```

```
                                        page
                                ;
                                ;       GROUP-LIST MESSAGE SUBTYPE
                                ;
                                ;       Add his group names to our table, if we are missing any of them.
                                ;       At the same time, construct a table to map his relative group
                                ;       numbers into ours, so that we can remap the groupmask in his
                                ;       station records to match ours.
                                ;
                                ;       di points to the groupname message subtype in the receive buffer,
                                ;       and is updated to point to the next message subtype when we're done.
                                ;
0599                            process_grp_msg proc    near

0599  B2 00                              mov     dl,0                    ;his relative group number
                                          $do                             ;repeat for all his groups

059B  FE C2                                       inc     dl                      ;increment his group number
059D  8A 0D                                       mov     cl,[di].grp_namel       ;length of his groupname
059F  0A C9                                       or      cl,cl
                                                  $exitif z                       ;done if zero
05A3  80 C1 01                                    add     cl,grp_rec_fsize;cx=size of his whole grp_rec
05A6  32 ED                                       xor     ch,ch

                                ;       Lookup group name to see if we have it already

05A8  8D 36 0B71 R                                lea     si,grp_list             ;start of our group list
05AC  B6 00                                       mov     dh,0                    ;our relative group number
                                                  $do                             ;repeat for all our groups
05AE  FE C6                                        inc    dh                      ;increment our group number
05B0  8A 1C                                        mov    bl,[si].grp_namel       ;length of our group name
05B2  0A DB                                        or     bl,bl
                                                   $exitif z                      ;no more: not found
05B6  57                                           push   di
05B7  56                                           push   si
05B8  51                                           push   cx
05B9  F3/ A6                                       repe   cmpsb                   ;compare names
05BB  59                                           pop    cx
05BC  5E                                           pop    si
05BD  5F                                           pop    di
05BE  74 2E                                        je     pr_grp_found            ;found
05C0  32 FF                                        xor    bh,bh
05C2  8D 70 01                                     lea    si,grp_rec_fsize[si+bx] ;skip to next group of ours
                                                  $repeat

                                ;       Add new group name to our list

05C7  57                                          push    di                      ;not found...
05C8  51                                          push    cx
05C9  01 0E 0B6E R                                add     grp_list_size,cx;add to our group list
05CD  81 3E 0B6E R 00C8                           cmp     grp_list_size,max_grp_list  ;(if it fits)
                                                  $if     b
```

```
05D5  FE 06 0B70 R                      inc   grp_count               ;one more group name
05D9  87 F7                             xchg  si,di
05DB  F3/ A4                            rep   movsb                   ;move the name
05DD  32 C0                             xor   al,al
05DF  AA                                stos  grp_list                ;add ending zero
05E0  80 0E 0911 R 04                   or    lists_chgd,ch_grp       ;flag grouplist changed
                                      $else
05E8  29 0E 0B6E R                      sub   grp_list_size,cx
                                      $endif
05EC  59                              pop     cx
05ED  5F                              pop     di

                              ;       Create group map such that grp_number_map[his_group] = our_group

05EE  8A DA                   pr_grp_found:   mov     bl,dl               ;his group number
05F0  32 FF                                   xor     bh,bh
05F2  88 B7 0C3E R                            mov     grp_number_map[bx],dh   ;map to our group number

05F6  03 F9                                   add     di,cx               ;move to his next group

                                      $repeat

                                      if assert                           ;fill out the rest of his
05FA  32 FF                                   xor     bh,bh               ;  group map with zero
05FC  8A DA                                   mov     bl,dl               ;his group number
                                              $do
05FE  80 FB 10                                  cmp   bl,16
                                                $exitif e
0603  FE C3                                     inc   bl
0605  C6 87 0C3E R 00                           mov   grp_number_map[bx],0
                                              $repeat
                                      endif
060C  47                                      inc     di                  ;point at next subtype

060D  C3                                      ret

060E                          process_grp_msg endp
```

```
                                page

                        ;
                        ;       STATION-LIST MESSAGE SUBTYPE
                        ;
                        ;       Add his known stations to our list.   He himself is not a rumor,
                        ;       but all the other stations he sends are considered to be rumors.
                        ;       Add any group membership he knows that we don't.
                        ;
                        ;       di points to the station-list part of the message and is updated
                        ;       to point after it.
                        ;
060E                    process_stn_msg proc    near

                            $do                                     ;for all his stn_rec's
060E  80 3D 00                  cmp     [di].stn_addr,0             ;0 stn addr is end-of-list
                                $if     e
0613  E9 0697 R                   jmp   proc_stn_end
                                $endif

                        ;       Remap his groupmap to our group numbering

0616  33 C9                             xor     cx,cx               ;remap this groupmask
0618  8B 45 02                          mov     ax,[di].stn_groupmask   ;  to our ordering
061B  33 D2                             xor     dx,dx               ;dx=new groupmask
                                        $do
061D  41                                  inc   cx                  ;his relative group number
061E  0A E4                               or    ah,ah
                                          $if   s                   ;it is a member of that group
0622  8B D9                                mov  bx,cx
0624  8A 9F 0C3E R                        mov   bl,grp_number_map[bx]   ;get our number for it
0628  D1 E3                               shl   bx,1
                                          if    assert
                                          $if   z
062C  B0 04                                mov al,4                 ;bad group bit in rcvd msg
062E  E8 0E0A R                            call int_error
                                          $endif
                                          endif
0631  0B 97 069C R                        or    dx,group_bits[bx]   ;turn on our bit for it
                                        $endif
0635  D1 E0                               shl   ax,1                ;next bit
0637  83 F9 10                           cmp   cx,16
                                        $repeatuntil    e
063C  89 55 02                          mov     [di].stn_groupmask,dx   ;store the translated map

                        ;       Look up the station in our list, and add it if necessary

063F  8A 1D                             mov     bl,[di].stn_addr;get the stn addr
0641  3A 1E 0C56 R                      cmp     bl,rcv_arcnet       ;he himself?

                                        $if     e                   ;sender:
0647  E8 0C68 R                           call  stn_lookup          ;look him up
                                          $if   z
```

```
064C  80 65 01 7F                                  and  [di].stn_flags,255-rumor;not found: add him as non-rumor
0650  E8 0C84 R                                    call stn_add
0653  8D 06 0C8B R                                 lea  ax,tweedle4              ;special tweedle for new users
0657  A3 0C7F R                                    mov  tweedle_ptr,ax
                                                  $else
065D  F6 44 01 80                                  test [si].stn_flags,rumor     ;found: was he a rumor before?
                                                   $ifnot z
0663  80 64 01 7F                                   and [si].stn_flags,255-rumor;yes: make him a non-rumor
0667  FE 0E 0915 R                                  dec rumor_count
066B  80 0E 0911 R 01                               or  lists_chgd,ch_stn        ;and flag the list as changed
                                                   $endif
0670  E8 06BE R                                    call chk_stn_changes          ;check for name/group changes
                                                  $endif

                                                 $else                           ;not sender:
0676  80 4D 01 80                                  or    [di].stn_flags,rumor    ;  treat as a rumor
067A  E8 0C68 R                                    call  stn_lookup              ;lookup that station
                                                   $if   z
067F  E8 0C84 R                                     call stn_add                 ;not found: add it as a rumor
                                                   $else
0685  8B 45 02                                      mov  ax,[di].stn_groupmask   ;found: add any new groups
0688  09 44 02                                      or   [si].stn_groupmask,ax
                                                   $endif
                                                 $endif

068B  8A 45 04                                   mov     al,[di].stn_namel       ;move past this entry
068E  83 C7 05                                   add     di,stn_rec_fsize
0691  98                                         cbw
0692  03 F8                                      add     di,ax
                                               $repeat                           ;for all his stations


0697                                 proc_stn_end:
0697  E8 0CF7 R                                 call    group_purge              ;purge any unused groups

069A  47                                         inc     di                       ;point at next subtype
069B  C3                                         ret

069C  0000 8000 4000 2000            group_bits  dw      0,8000h,4000h,2000h,1000h,0800h,0400h,0200h,0100h
      1000 0800 0400 0200
      0100
06AE  0080 0040 0020 0010                        dw        0080h,0040h,0020h,0010h,0008h,0004h,0002h,0001h
      0008 0004 0002 0001

06BE                                 process_stn_msg endp
```

```
                                page

                        ;
                        ; chk_stn_changes       Check for any name or group membership changes for
                        ;                       the station from whom we just received a msg.
                        ;                       Adjust (and/or move) the stn_list record if so.
                        ;
                        ;       Input:  si points to our stn_list record for him
                        ;               di points to his stn_list record for himself
                        ;
                        ;       Destroys: ax, bx, cx, si
                        ;                 (si is not really destroyed, but the stn_list may be
                        ;                  reordered, so si might not point at what it used to.
                        ;

06BE                    chk_stn_changes proc    near
06BE  8B 45 02                  mov     ax,[di].stn_groupmask   ;any group changes?
06C1  3B 44 02                  cmp     ax,[si].stn_groupmask
                                $ifnot e                        ;yes: use new groups
06C6  89 44 02                   mov    [si].stn_groupmask,ax
06C9  C6 06 0911 R 02            mov    lists_chgd,ch_stn_atr
                                $endif
06CE  8A 4C 04                  mov     cl,[si].stn_namel       ;compare names
06D1  FE C1                     inc     cl                      ; (and namelengths)
06D3  32 ED                     xor     ch,ch
06D5  33 DB                     xor     bx,bx
                                $do
06D7  8A 40 04                    mov   al,stn_name1[si+bx]
06DA  3A 41 04                    cmp   al,stn_name1[di+bx]
                                  $ifnot e                      ;different: he changed his name!
06DF  F6 06 001C R 01              test send_busy,1             ;rearrange stations only if
                                   $if  z                       ;  send task is not using it!
06E6  8A 44 01                      mov al,stn_flags[si];copy our flags for him
06E9  88 45 01                      mov stn_flags[di],al
06EC  E8 0CC0 R                     call stn_remove             ;remove and re-add the stn_rec
06EF  E8 0C84 R                     call stn_add
06F2  C6 06 0C50 R 01               mov dest_stn,1              ;(hack!)
                                   $endif
06F7  EB 03                        jmp short chk_stn_ch_xit
                                  $endif
06F9  43                          inc   bx
                                $repeatloop
06FC  C3                chk_stn_ch_xit: ret

06FD                    chk_stn_changes endp
```

```
                                        page

                                ;
                                ;       TEXT MESSAGE SUBTYPE
                                ;
                                ;       Process the text part of the message, which is the form:
                                ;
                                ;          from <right_arrow >to: text... <CR>
                                ;
06FD                            process_txt_msg proc    near

06FD  80 3D 0D                          cmp     byte ptr [di],cr;don't do null text messages

                                    $ifnot      e


                                ;       Make the appropriate noise, and force window popup sometimes

0702  F6 06 0022 R 01                   test    windows_open,1
                                        $ifnot  z
0709  8D 06 0C89 R                       lea   ax,tweedle3             ;if windows are open
                                        $else
0710  F6 06 002B R 01                    test  auto_popup,1
                                         $ifnot z
0717  C6 06 0023 R 01                     mov do_open,1                ;if windows are to popup
071C  C6 06 0029 R 01                     mov popped_up,1
0721  8D 06 0C85 R                       lea ax,tweedle2
                                        $else
0728  8D 06 0C81 R                       lea ax,tweedle1               ;if windows stay closed
                                        $endif
                                        $endif
072C  A3 0C7F R                         mov     tweedle_ptr,ax


072F  B8 0601                           mov     ax,select_dwin          ;select display window
0732  E8 0000 E                        call    window
0735  E8 0DC9 R                        call    do_timestamp            ;print timestamp, maybe
0738  B4 08                            mov     ah,window_setattr
073A  A0 0033 R                         mov     al,char_bold            ;set bold characters
073D  E8 0000 E                         call    window

                                        $do                             ;now the message
0740  8A 05                              mov   al,byte ptr [di];get a character
0742  47                                 inc   di
0743  3C 0D                             cmp   al,cr
                                         $exitif e                      ;stop at cr
0747  E8 0D92 R                          call  w_print                  ;display it
074A  3C 3A                              cmp   al,':'                   ;was it the colon?
                                         $if   e                        ;yes: switch to normal chars
074E  B4 08                               mov  ah,window_setattr
0750  A0 0032 R                           mov  al,char_normal
0753  E8 0000 E                           call window
                                         $endif
```

```
                                        $repeat
0758  B4 08                              mov     ah,window_setattr
075A  A0 0032 R                         mov     al,char_normal
075D  E8 0000 E                         call    window
0760  B8 0602                           mov     ax,select_twin          ;reselect typing window
0763  E8 0000 E                         call    window

                                      $endif                            ;null message check

0766  C3                                ret

0767                          process_txt_msg endp

                                        assume  ds:nothing,es:nothing
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;
                        ; recv_message         Receive a message into rcv_buffer.
                        ;                      Sets rcv_arcnet and  rcv_msg_type.
                        ;                      Destroys ax.
                        ;
                        ;-----------------------------------------------------------------------------

0767                    recv_message    proc    near
0767  1E                              push    ds
0768  56                              push    si
0769  2E: C5 36 0C5E R               lds     si,rcv_rb               ;get rb in ds:si

076E                    recv_wait:
                                      $do
076E  E8 0E69 R                         call  dispatch
0771  C7 44 40 017D R                   mov   [si].recv_ptr, offset cgrp:rcv_buffer   ;buffer addr
0776  8C C8                             mov   ax,cs
0778  89 44 42                          mov   [si].recv_ptr+2,ax
077B  C7 44 44 03C3                     mov   [si].recv_limit,rcv_bufsize             ;buffer size
0780  2E: A1 0C5C R                     mov   ax,wks
0784  89 44 30                          mov   [si].hdr_src_socket,ax                  ;listen socket
0787  2E: F6 06 0026 R 01               test  open_recvd,1            ;recv already true from kb int?
                                        $exitif nz                    ;yes: don't call again
                                        14_call open_recv

                                      $repeatuntil c                  ;wait for incoming message
0796  2E: C6 06 0026 R 01             mov     open_recvd,1            ;signal message incoming
079C  2E: C6 06 001D R 01             mov     recv_busy,1             ;and our task busy

                                      $do                             ;watchout for race condition:
07A2  2E: F6 06 0024 R 01               test  do_stop,1               ; keyboard may already be trying
                                        $exitif z                     ; to shutdown the task dispatcher
07AA  E8 0E69 R                         call  dispatch
                                      $repeat

07AF  2E: C6 06 0026 R 00             mov     open_recvd,0            ;clear open_recv pending flag

07B5  80 7C 55 00                     cmp     [si].his_bcst,0         ;if it wasn't a broadcast,
                                      $if     z                       ;  then acknowledge it now
                                      14_call ack_now
                                      $do                             ;churn until ack done
07C0  E8 0E69 R                        call   dispatch
                                       14_call 14churn
07C8  8A 44 3E                         mov    al,[si].recv_status
07CB  3C 05                            cmp    al,tf_in_prog
                                      $repeatwhile e
07CF  3C 00                           cmp     al,tf_idle              ;should be idle
07D1  75 18                           jne     rcv_abort
                                      $endif

                                      14_call recv_msg                ;receive it
```

```
                                        $do
07D8  E8 0E69 R                         call    dispatch
                                        l4_call l4churn                 ;churn until receive done
07E0  8A 44 3E                          mov     al,[si].recv_status
07E3  3C 05                             cmp     al,tf_in_prog
                                        $repeatwhile e

07E7  3C 00                             cmp     al,tf_idle              ;should be idle now
                                        $ifnot  e
07EB                            rcv_abort:       l4_call abort_conn     ;receive error: abort connection
07F0  2E: C6 06 001D R 00       rcv_try_again:   mov    recv_busy,0     ;  signal nothing in progress
07F6  E9 076E R                          jmp    recv_wait               ;  and keep waiting
                                        $endif

07F9  8A 44 54                          mov     al,[si].his_arc         ;save his arcnet address
07FC  2E: A2 0C56 R                     mov     rcv_arcnet,al
0800  8A 44 3F                          mov     al,[si].recv_type       ;save message type
0803  2E: A2 0C5A R                     mov     rcv_msg_type,al

0807  2E: 80 3E 0C5A R 96               cmp     rcv_msg_type,rqscrn_msg_type
080D  74 3C                             je      send_screen             ;special request type
080F  2E: 80 3E 0C5A R 94               cmp     rcv_msg_type,kbchar_msg_type
0815  74 7B                             je      rcv_key                 ;special request type

0817  2E: 80 3E 0C5A R 98               cmp     rcv_msg_type,logon_msg_type
                                        $if     e
                                         l4_call abort_conn             ;abort broadcast messages
                                        $else
0827  E8 082D R                          call   rcv_disconn             ;disconnect all others
                                        $endif

082A  5E                                pop     si
082B  1F                                pop     ds
082C  C3                                ret


082D                            rcv_disconn:     l4_call disconn        ;disconnect
                                        $do
0832  E8 0E69 R                           call  dispatch
                                          l4_call l4churn               ;churn until not connected
083A  8A 44 08                            mov   al,[si].conn_status
083D  3C 00                               cmp   al,cn_established
                                        $repeatwhile e
0841  3C 01                             cmp     al,cn_not_conn          ;check for errors
                                        $ifnot  e
                                          l4_call abort_conn            ;and abort if so
                                        $endif
084A  C3                                ret


084B  C6 44 33 95               send_screen:     mov     [si].hdr_data_type,scrn_msg_type
084F  C7 44 0A 0000                              mov     [si].send_ptr,0
0854  C7 44 0C B000                              mov     [si].send_ptr+2,0b000h  ;assume b/w screen
```

```
0859  CD 11                                int     equip_int               ;equipment determination
085B  24 30                                and     al,30h                  ;00 = no card, use b/w
                                           $ifnot  z
085F  3C 30                                 cmp    al,30h                  ;11 = b/w screen
                                            $ifnot z
0863  C7 44 0C B800                         mov    [si].send_ptr+2,0b800h  ;01,10 = color screen
                                            $endif
                                           $endif
0868  C7 44 0E 0FA0                        mov     [si].send_length,25*80*2
                                           l4_call send_msg
0872  E8 0E69 R                send_scr_lp: call    dispatch
                                           l4_call l4churn
087A  8A 44 09                             mov     al,[si].send_status
087D  3C 05                                cmp     al,tf_in_prog
087F  74 F1                                  je    send_scr_lp
0881  3C 06                                cmp     al,tf_ack_wait
0883  74 ED                                  je    send_scr_lp
0885  3C 00                                cmp     al,tf_idle
                                           $ifnot  e
0889  E9 07EB R                              jmp   rcv_abort
                                           $endif
088C  E8 082D R                            call    rcv_disconn             ;disconnect
088F  E9 07F0 R                            jmp     rcv_try_again           ;and wait for another msg


= 001A                         kb_buffer_head  equ     1ah
= 001C                         kb_buffer_tail  equ     1ch
= 0080                         kb_buffer_start equ     80h
= 0082                         kb_buffer_end   equ     82h

0892  06                       rcv_key:        push    es
0893  B8 0040                               mov     ax,40h                  ;bios data area
0896  8E C0                                 mov     es,ax
0898  FA                                    cli                             ;disable ints
0899  26: 8B 1E 001C                        mov     bx,es:kb_buffer_tail
089E  83 C3 02                             add     bx,2                    ;increment kb pointer
08A1  26: 3B 1E 0082                       cmp     bx,es:kb_buffer_end
                                           $if     e                       ;circularly
08A8  26: 8B 1E 0080                         mov   bx,es:kb_buffer_start
                                           $endif
08AD  26: 3B 1E 001A                       cmp     bx,es:kb_buffer_head
                                           $ifnot  e                       ;no buffer overflow
08B4  26: 87 1E 001C                         xchg  bx,es:kb_buffer_tail    ;store new ptr, get old
08B9  2E: A1 017D R                          mov   ax,word ptr cs:rcv_buffer ;get 1st char/scancode from buffer
08BD  26: 89 07                              mov   es:[bx],ax               ;store char/scancode in buffer
                                           $else
                                                 ;buffer ovfl. Do what?
                                           $endif
08C3  FB                                   sti                             ;enable ints
08C4  07                                   pop     es
08C5  E8 082D R                            call    rcv_disconn             ;disconnect
08C8  E9 07F0 R                            jmp     rcv_try_again           ;and wait for another msg

08CB                           recv_message    endp
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                        ;
                        ; Network send task
                        ;
                        ;       If the keyboard buffer is full, send a queued text line to all stations,
                        ;       along with our list of known groups and stations.
                        ;
                        ;       If a broadcast response was queued, send only the station list.
                        ;
                        ;       If a send_msg request was queued from the kb_int function, send
                        ;       the buffer supplied.
                        ;
                        ;       If the rumor_count is non-zero, send test probes to determine
                        ;       the status of the rumors.
                        ;
                        ;-----------------------------------------------------------------------------
                        ;-----------------------------------------------------------------------------
                                %out    -----send_task

08CB                    send_task       proc    near
08CB  8C C8                             mov     ax,cs
08CD  8E D8                             mov     ds,ax           ;ds:=cs
08CF  8E C0                             mov     es,ax           ;es:=cs
                                        assume  ds:cgrp,es:cgrp


                        ; But first: Send a one-time-only broadcast message requesting station lists
                        ; from anyone out there.

08D1  C6 06 001C R 01                   mov     send_busy,1             ;we are busy
08D6  E8 0B36 R                         call    copy_lists              ;make up the station list (only us)
08D9  32 C0                             xor     al,al
08DB  AA                                stos    snd_buffer              ;end of subtypes
08DC  81 EF 0540 R                      sub     di,offset cgrp:snd_buffer  ;length of message
08E0  89 3E 0908 R                      mov     snd_buffer_size,di
08E4  C6 06 0C57 R FF                   mov     snd_arcnet,0ffh         ;broadcast destination
08E9  B4 98                             mov     ah,logon_msg_type       ;logon message type
08EB  E8 0BA9 R                         call    send_message            ;send it
08EE  C6 06 001C R 00                   mov     send_busy,0             ;we aren't busy any more


                                $do     ;**************        ;start send task infinite loop
08F3  E8 0E69 R                         call    dispatch


08F6  F6 06 0054 R 01                   test    kbd_buffer_full,1       ;keyboard buffer full?
                                        $ifnot  z
08FD  E8 0995 R                           call  send_kb_msg             ;yes: send keyboard message
                                        $endif
```

```
0900  80 3E 0C58 R 00                cmp     brdcst_arcnet,0         ;broadcast response queued?
                                     $ifnot  z
0907  C6 06 001C R 01                  mov   send_busy,1             ;we are busy
090C  E8 0B36 R                        call  copy_lists              ;make up the station list
090F  32 C0                            xor   al,al
0911  AA                               stos  snd_buffer              ;end of subtypes
0912  81 EF 0540 R                     sub   di,offset cgrp:snd_buffer  ;length of message
0916  89 3E 0908 R                     mov   snd_buffer_size,di
091A  B0 00                            mov   al,0
091C  86 06 0C58 R                     xchg  al,brdcst_arcnet;get station addr, reset queue
0920  A2 0C57 R                        mov   snd_arcnet,al
0923  B4 99                            mov   ah,genl_msg_type
0925  80 0E 0911 R 02                  or    lists_chgd,ch_stn_atr   ;force underline
092A  E8 0BA9 R                        call  send_message            ;send the response
092D  80 0E 0911 R 02                  or    lists_chgd,ch_stn_atr   ;remove underline
0932  C6 06 001C R 00                  mov   send_busy,0             ;we are not busy
                                     $endif


0937  80 3E 0027 R 00                cmp     send_msg_rqst,0         ;kb_int send-msg request?
                                     $ifnot  z
093E  C6 06 001C R 01                  mov   send_busy,1             ;yes: make this task busy
0943  C6 06 0027 R 00                  mov   send_msg_rqst,0         ;clear request
0948  8D 3E 0540 R                     lea   di,snd_buffer
094C  B0 CC                            mov   al,txt_subtype          ;send text-only message
094E  AA                               stos  snd_buffer
094F  1E                               push  ds
                                       assume ds:nothing
0950  2E: C5 36 0057 R                 lds   si,send_msg_ptr         ;ptr given by caller to kb_int
                                       $do
0955  AC                                lodsb
0956  AA                                stos snd_buffer              ;copy to output buffer
0957  3C 0D                             cmp  al,cr                   ;until CR
                                       $repeatuntil e
095B  1F                               pop   ds
                                       assume ds:cgrp
095C  32 C0                            xor   al,al
095E  AA                               stos  snd_buffer              ;end of subtypes
095F  81 EF 0540 R                     sub   di,offset cgrp:snd_buffer  ;length of message
0963  89 3E 0908 R                     mov   snd_buffer_size,di
0967  A0 005B R                        mov   al,send_msg_stn         ;station to send to
096A  A2 0C57 R                        mov   snd_arcnet,al
096D  B4 99                            mov   ah,genl_msg_type
096F  E8 0BA9 R                        call  send_message            ;send it
                                       $if   c
0974  C6 06 005C R 01                   mov  send_msg_stat,1         ;failed: change status
                                       $endif
0979  C6 06 001C R 00                   mov  send_busy,0
                                     $endif


097E  80 3E 0915 R 00                cmp     rumor_count,0           ;any rumors to verify
                                     $ifnot  e                       ;yes
0985  F6 06 0021 R 01                  test  other_conn_busy,1       ;other rb's active?
```

```
                                        $if   z                         ;no - can take time for this
098C  E8 0AC1 R                          call probe_rumors              ;determine rumor status
098F  E8 0CF7 R                          call group_purge               ;purge groups that became empty
                                        $endif
                                      $endif


                                $repeat ;**************         ;end of send task infinite loop

0995                          send_task       endp
```

```
                                page

                        ;
                        ;       Construct and send a message to all known and rumored stations
                        ;
0995                    send_kb_msg     proc    near

                        ;       Copy the group-list and part of the station-list

0995  E8 0B36 R                        call    copy_lists              ;copy the station namelist

0998  B0 CC                             mov     al,txt_subtype          ;start text subtype
099A  AA                                stos    snd_buffer
099B  89 3E 090A R                      mov     snd_buffer_txt,di       ;save offset of text part


                        ;       Construct from-->to: header

099F  32 ED                             xor     ch,ch
09A1  8A 0E 091A R                      mov     cl,stn_list.stn_namel   ;length of our name
09A5  80 F9 08                          cmp     cl,8
                                        $if     g
09AA  B1 08                               mov   cl,8                    ;(max 8)
                                        $endif
09AC  8D 36 091B R                      lea     si,stn_list.stn_name
09B0  F3/ A4                            rep     movsb                   ;move it
09B2  B0 1A                             mov     al,26                   ;little right-arrow character
09B4  AA                                stos    snd_buffer
09B5  E8 0B6E R                         call    get_dest_rec            ;get destination namel in si
09B8  AC                                lodsb                           ;get length
09B9  8A C8                             mov     cl,al
09BB  80 F9 08                          cmp     cl,8
                                        $if     g
09C0  B1 08                               mov   cl,8                    ;(max 8)
                                        $endif
09C2  F3/ A4                            rep     movsb                   ;move it
09C4  B0 3A                             mov     al,":"
09C6  AA                                stos    snd_buffer              ;and colon


                        ;       Move keyboard text

09C7  8D 36 005F R                      lea     si,kbd_buffer           ;move keyboard buffer
                                        $do
09CB  AC                                  lods  kbd_buffer
09CC  AA                                  stos  snd_buffer
09CD  3C 0D                               cmp   al,cr                   ;until cr
                                        $repeatuntil e
09D1  32 C0                             xor     al,al
09D3  AA                                stos    snd_buffer              ;ending zero
09D4  81 EF 0540 R                      sub     di,offset cgrp:snd_buffer  ;length of message
09D8  89 3E 0908 R                      mov     snd_buffer_size,di
```

```
09DC  C6 06 001C R 01                   mov     send_busy,1             ;signal that we are busy
09E1  C6 06 0054 R 00                   mov     kbd_buffer_full,0       ;free the keyboard task
09E6  8D 36 0916 R                      lea     si,stn_list             ;point to station list


                              ;       Send to the destination stations

09EA  C6 06 0C52 R 00                   mov     send_ok_count,0         ;count of sucessful sends
09EF  C6 06 0C53 R 00                   mov     send_ng_count,0         ;count of unsucessful sends
09F4  8A 0E 0C4F R                      mov     cl,dest_grp             ;compute dx=group mask, if any
09F8  33 D2                             xor     dx,dx
09FA  F9                                stc
09FB  D3 DA                             rcr     dx,cl

09FD  8D 36 0916 R                      lea     si,stn_list             ;walk down station list
0A01  B1 01                             mov     cl,1                    ;and count them
                                    $do                                 ;for all stations

0A03  F6 06 0C51 R 01                   test    is_dest_stn,1           ;what kind of destination?
                                        $ifnot z                        ;station...
0A0A  3A 0E 0C50 R                        cmp   cl,dest_stn             ;yes: right one?
0A0E  75 59                               jne   send_next               ;  no: skip to next
0A10  EB 05                               jmp   short send_it           ;  yes: send it
                                        $endif
0A12  85 54 02                          test    dx,[si].stn_groupmask   ;group...
0A15  74 52                             jz      send_next               ;not member

0A17  8A 04                     send_it:        mov     al,[si].stn_addr;get station address
0A19  3A 06 0C59 R                      cmp     al,our_arcnet
                                        $if     e                       ;don't send to ourself
0A1F  EB 48                               jmp   short send_next         ; (do it with $if - I dare you!)
                                        $endif
0A21  A2 0C57 R                         mov     snd_arcnet,al
0A24  B4 99                             mov     ah,genl_msg_type;use text message type
0A26  80 0E 0911 R 02                   or      lists_chgd,ch_stn_atr   ;force underline
0A2B  E8 0BA9 R                         call    send_message            ;try to send it
0A2E  9C                                pushf
0A2F  80 0E 0911 R 02                   or      lists_chgd,ch_stn_atr   ;remove underline
0A34  9D                                popf
                                        $if     c                       ;failed:
0A37  FE 06 0C53 R                        inc   send_ng_count           ;   count it
0A3B  8D 06 0C8F R                        lea   ax,tweedle5             ;   speaker burp
0A3F  A3 0C7F R                           mov   tweedle_ptr,ax
0A42  E8 0CC0 R                           call  stn_remove              ;   remove this station
0A45  38 0E 0C50 R                        cmp   dest_stn,cl             ;   adjust if destination is it or later
                                          $if   ge
0A4B  FE 0E 0C50 R                         dec  dest_stn
                                          $endif
                                        $else                           ;succeeded:
0A52  FE 06 0C52 R                        inc   send_ok_count           ;   count it
0A56  F6 44 01 80                         test  [si].stn_flags,rumor    ;   was he a rumor?
                                          $ifnot z
0A5C  80 64 01 7F                           and [si].stn_flags,255-rumor;  yes: not any more!
```

```
0A60  FE 0E 0915 R                        dec rumor_count
0A64  80 0E 0911 R 01                     or  lists_chgd,ch_stn       ;  stationlist is changed
                                        $endif

0A69  8A 44 04                send_next:  mov   al,[si].stn_namel      ;skip to next stn_rec
0A6C  98                                  cbw
0A6D  03 F0                               add   si,ax
0A6F  83 C6 05                            add   si,stn_rec_fsize
0A72  FE C1                               inc   cl                    ;next station number
                                        $endif                        ;fail check

0A74  80 3C 00                            cmp   [si].stn_addr,0       ;continue 'til end of stn list
                                    $repeatuntil e

                              ;     If any sends failed and stations were removed from our list,
                              ;     check if any group names are now unused and may be removed.

0A79  80 3E 0C53 R 00                     cmp   send_ng_count,0
                                        $if   g
0A80  E8 0CF7 R                           call  group_purge
                                        $endif

                              ;     Print it in our window, if it was sent to at least one

0A83  80 3E 0C52 R 00                     cmp   send_ok_count,0
                                    $if   g
0A8A  B8 0601                             mov   ax,select_dwin
0A8D  E8 0000 E                           call  window                ;now print it in our display window
0A90  E8 0DC9 R                           call  do_timestamp          ;print timestamp, maybe
0A93  B4 08                               mov   ah,window_setattr
0A95  A0 0033 R                           mov   al,char_bold          ;set bold characters
0A98  E8 0000 E                           call  window
0A9B  8B 36 090A R                        mov   si,snd_buffer_txt     ;start of text part of msg
                                          $do
0A9F  AC                                    lods  snd_buffer          ;print the message
0AA0  3C 0D                                 cmp   al,cr
                                            $exitif e                 ;stop at cr
0AA4  E8 0D92 R                             call  w_print
0AA7  3C 3A                                 cmp   al,":"              ;stop bold after colon
                                            $if   e
0AAB  B4 08                                   mov  ah,window_setattr
0AAD  A0 0032 R                               mov  al,char_normal     ;normal characters
0AB0  E8 0000 E                               call window
                                            $endif
                                          $repeat
0AB5  B8 0602                             mov   ax,select_twin        ;return to typing window
0AB8  E8 0000 E                           call  window
                                    $endif

0ABB  C6 06 001C R 00                     mov   send_busy,0           ;signal we are not busy
0AC0  C3                                  ret
0AC1                          send_kb_msg endp
```

```
                                        page
                                ;-----------------------------------------------------------------------------
                                ;
                                ; probe_rumors  Send a silent message to each rumored station to find
                                ;               out whether or not it really exists.
                                ;
                                ;-----------------------------------------------------------------------------

0AC1                            probe_rumors    proc    near
0AC1  C6 06 001C R 01                           mov     send_busy,1
0AC6  C6 06 0540 R 00                           mov     snd_buffer,0            ;null message
0ACB  C7 06 0908 R 0001                         mov     snd_buffer_size,1       ;  is a single zero
0AD1  8D 36 0916 R                              lea     si,stn_list             ;walk station list
0AD5  B1 01                                     mov     cl,1                    ;and count them
                                        $do
0AD7  F6 44 01 80                               test    [si].stn_flags,rumor    ;a rumor?
                                                $if     z
0ADD  EB 3C                                       jmp   short probe_next;no - skip to next
                                                $endif
0ADF  8A 04                                     mov     al,[si].stn_addr;yes: send to it
0AE1  A2 0C57 R                                 mov     snd_arcnet,al
0AE4  B4 99                                     mov     ah,genl_msg_type
0AE6  80 0E 0911 R 02                           or      lists_chgd,ch_stn_atr   ;force underline
0AEB  E8 0BA9 R                                 call    send_message            ;send
0AEE  9C                                        pushf
0AEF  80 0E 0911 R 02                           or      lists_chgd,ch_stn_atr   ;remove underline
0AF4  9D                                        popf
                                                $if     c                       ;failed:
0AF7  8D 06 0C8F R                                lea   ax,tweedle5             ;  speaker burp
0AFB  A3 0C7F R                                   mov   tweedle_ptr,ax
0AFE  E8 0CC0 R                                   call  stn_remove              ;  remove it
0B01  38 0E 0C50 R                                cmp   dest_stn,cl             ;  adjust destination if this or later
                                                  $if   ge
0B07  FE 0E 0C50 R                                 dec  dest_stn
                                                  $endif
                                                $else                           ;succeeded:
0B0E  80 64 01 7F                                 and   [si].stn_flags,255-rumor;  make him a non-rumor
0B12  FE 0E 0915 R                                dec   rumor_count
0B16  80 0E 0911 R 01                             or    lists_chgd,ch_stn       ;  stationlist is changed
0B1B  8A 44 04                  probe_next:       mov   al,[si].stn_namel       ;  skip to next station
0B1E  98                                          cbw
0B1F  03 F0                                       add   si,ax
0B21  83 C6 05                                    add   si,stn_rec_fsize
0B24  FE C1                                       inc   cl                      ;next station number
                                                $endif

0B26  80 3C 00                                  cmp     [si].stn_addr,0         ;continue 'til end of stn list
                                        $repeatuntil e
                                                ;assert rumor_count=0 already!
0B2B  C6 06 0915 R 00                           mov     rumor_count,0
0B30  C6 06 001C R 00                           mov     send_busy,0
0B35  C3                                        ret
0B36                            probe_rumors    endp
```

```
                        page
                        ;-----------------------------------------------------------------------
                        ;
                        ; copy_lists   Copy our group name list to the output buffer.
                        ;              Then copy a subset of our station list to the snd_buffer,
                        ;              including only non-rumored stations (stations that we have
                        ;              successfully communicated with).
                        ;
                        ;              Returns with offset into snd_buffer in di.
                        ;              Destroys ax, cx, si.
                        ;
                        ;-----------------------------------------------------------------------

0B36                    copy_lists      proc    near
0B36  8D 3E 0540 R                      lea     di,snd_buffer           ;index into send buffer

0B3A  B0 BB                             mov     al,grp_subtype          ;grp_list message subtype
0B3C  AA                                stos    snd_buffer
0B3D  8D 36 0B71 R                      lea     si,grp_list
0B41  8B 0E 0B6E R                      mov     cx,grp_list_size
0B45  F3/ A4                            rep     movsb                           ;move it

0B47  B0 AA                             mov     al,stn_subtype          ;stn_list message subtype
0B49  AA                                stos    snd_buffer
0B4A  8D 36 0916 R                      lea     si,stn_list             ;start of our station list
0B4E  B5 00                             mov     ch,0                    ;prepare for counting
                                $do                                     ;loop for all stations
0B50  F6 04 FF                          test    [si].stn_addr,0ffh
                                        $exitif z                       ;end of list
0B55  8A 4C 04                          mov     cl,[si].stn_namel       ;compute length of this rec
0B58  83 C1 05                          add     cx,stn_rec_fsize
0B5B  F6 44 01 80                       test    [si].stn_flags,rumor    ;rumor?
                                        $ifnot  z
0B61  03 F1                               add   si,cx                   ;rumor: skip it
                                        $else
0B66  F3/ A4                              rep   movsb                   ;not rumor: copy it
                                        $endif
                                $repeat                                 ;all stations
0B6A  B0 00                             mov     al,0
0B6C  AA                                stosb                           ;zero to mark end of list
0B6D  C3                                ret
0B6E                    copy_lists      endp
```

```
                                page
                        ;-----------------------------------------------------------------------------
                        ;
                        ; get_dest_rec  Get the destination record (either grp_rec or stn_rec)
                        ;               of the current destination.
                        ;
                        ;               Return si pointing to {name_length,"name"} entry of record.
                        ;               Destroys: ax, bx, cl
                        ;
                        ;-----------------------------------------------------------------------------

0B6E                    get_dest_rec    proc    near
0B6E  32 FF                             xor     bh,bh
0B70  B1 01                             mov     cl,1                    ;station/group counter

0B72  F6 06 0C51 R 01                   test    is_dest_stn,1
                                        $ifnot  z
0B79  8D 36 0916 R                        lea   si,stn_list             ;station destination
                                          $do
0B7D  3A 0E 0C50 R                         cmp  cl,dest_stn
                                           $exitif e                    ;got it...
0B83  8A 5C 04                             mov  bl,[si].stn_namel
0B86  8D 70 05                             lea  si,stn_rec_fsize[si+bx] ;move to next
0B89  FE C1                                inc  cl
                                          $repeat
0B8D  8D 74 04                            lea   si,stn_namel[si];return ptr to name_length
                                                                 .
                                        $else
0B93  8D 36 0B71 R                        lea   si,grp_list             ;group destination
                                          $do
0B97  3A 0E 0C4F R                         cmp  cl,dest_grp
                                           $exitif e                    ;got it...
0B9D  8A 1C                                mov  bl,[si].grp_namel
0B9F  8D 70 01                             lea  si,grp_rec_fsize[si+bx] ;move to next
0BA2  FE C1                                inc  cl
                                          $repeat
0BA6  8D 34                               lea   si,grp_namel[si];return ptr to name_length
                                        $endif

0BA8  C3                                ret
0BA9                    get_dest_rec    endp

                                        assume  ds:nothing,es:nothing
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ; send_message         Send the snd_buffer, size snd_buffer_size
                        ;                     to the station snd_arcnet. (ff to broadcast).
                        ;                     Message type is in ah.
                        ;                     If not broadcast, return C if it fails, otherwise NC.
                        ;                     Destroys ax, bx.
                        ;
                        ;-----------------------------------------------------------------------

0BA9                    send_message    proc    near
0BA9  1E                                push    ds
0BAA  56                                push    si

0BAB  2E: C5 36 0C66 R                  lds     si,l4_publics
0BB0  BB 0006                           mov     bx,timeout              ;set new short level2 timeouts
0BB3  87 5C 22                          xchg    bx,ds:[si].to_pkt_keep  ;(but save old values on the stack)
0BB6  53                                push    bx
0BB7  BB 0006                           mov     bx,timeout
0BBA  87 5C 24                          xchg    bx,ds:[si].to_ta_wait
0BBD  53                                push    bx

0BBE  2E: C5 36 0C62 R                  lds     si,snd_rb                       ;get rb in ds:si
0BC3  2E: A0 0C57 R                     mov     al,snd_arcnet
0BC7  88 44 54                          mov     [si].his_arc,al                 ;arcnet address
0BCA  88 64 33                          mov     [si].hdr_data_type,ah           ;type
0BCD  2E: 8B 1E 0908 R                  mov     bx,snd_buffer_size
0BD2  89 5C 0E                          mov     [si].send_length,bx             ;message length

0BD5  2E: 80 3E 0C57 R FF               cmp     snd_arcnet,0ffh
                                        $if     e                               ;broadcast:
0BDD  98                                  cbw                                   ;  address ffffff
0BDE  89 44 1E                            mov   [si].hdr_dest_host+0,ax
0BE1  89 44 20                            mov   [si].hdr_dest_host+2,ax
0BE4  89 44 22                            mov   [si].hdr_dest_host+4,ax
                                        $else
0BEA  C7 44 1E 0008                       mov   [si].hdr_dest_host+0,0008h      ;ethernet address
0BEF  C7 44 20 000A                       mov   [si].hdr_dest_host+2,000ah      ; 08000A0000aa
0BF4  C6 44 22 00                         mov   byte ptr[si].hdr_dest_host+4,0
0BF8  88 44 23                            mov   byte ptr[si].hdr_dest_host+5,al ;except low byte is arc
                                        $endif
0BFB  C7 44 0A 0540 R                   mov     [si].send_ptr,offset cgrp:snd_buffer    ;buffer address
0C00  8C C8                             mov     ax,cs
0C02  89 44 0C                          mov     [si].send_ptr+2,ax
0C05  2E: A1 0C5C R                     mov     ax,wks
0C09  89 44 24                          mov     [si].hdr_dest_socket,ax ;socket number

                                        l4_call connect                 ;try to connect and send

0C11  E8 0E69 R                 conn_wait:      call    dispatch        ;wait for connection
                                        l4_call l4churn
0C19  8A 44 09                          mov     al,[si].send_status
```

```
0C1C  3C 05                              cmp     al,tf_in_prog
0C1E  74 F1                                je    conn_wait
0C20  2E: 80 3E 0C57 R FF                cmp     snd_arcnet,0ffh        ;if broadcast,
0C26  74 20                                je    send_abort             ;abort now
0C28  3C 06                              cmp     al,tf_ack_wait
0C2A  74 E5                                je    conn_wait

0C2C  3C 00                              cmp     al,tf_idle
0C2E  75 18                                jne   send_abort             ;abort if not ok

                                         l4_call disconn                ;ok: disconnect

                                         $do
0C35  E8 0E69 R                            call  dispatch               ;wait for disconnect
                                           l4_call l4churn
0C3D  8A 44 09                             mov   al,[si].send_status
0C40  3C 00                                cmp   al,cn_established
                                         $repeatwhile e

0C44  3C 01                              cmp     al,cn_not_conn         ;disconnect ok?
                                         $ifnot  e
0C48                           send_abort:    l4_call abort_conn       ;no: abort the connection
0C4D  F9                                    stc                         ;return with C set
                                         $else
0C51  F8                                    clc                         ;ok: return with NC set
                                         $endif

0C52  2E: C5 36 0C66 R                   lds     si,l4_publics
0C57  5B                                 pop     bx
0C58  89 5C 24                           mov     ds:[si].to_ta_wait,bx  ;restore level2 timeouts
0C5B  5B                                 pop     bx                     ; (PRESERVE C/NC!)
0C5C  89 5C 22                           mov     ds:[si].to_pkt_keep,bx

0C5F  2E: C6 06 0C57 R 00                mov     snd_arcnet,0           ;clear out destination
                                                                        ;  for show_stations
0C65  5E                                 pop     si
0C66  1F                                 pop     ds
0C67  C3                                 ret
0C68                           send_message  endp
```

```
                                page

                        ;-----------------------------------------------------------------------
                        ;
                        ;       Station list routines.   Called by both send and receive tasks.
                        ;
                        ;       These require ds=es=cs!   You have been warned!
                        ;
                        ;-----------------------------------------------------------------------

                                assume  ds:cgrp,es:cgrp

                        ;
                        ; stn_lookup    Lookup station address bl.
                        ;               Return NZ and offset to stn_rec in si if found.
                        ;               Return Z if not found.
                        ;               Destroys ax.
                        ;

0C68                    stn_lookup      proc    near
0C68  8D 36 0916 R                      lea     si,stn_list             ;start of station list

                                        $do
0C6C  8A 04                               mov   al,[si].stn_addr;get station addr
0C6E  0A C0                               or    al,al                   ;end?
0C70  74 11                               jz    stn_not_found           ;yes...
0C72  3A C3                               cmp   al,bl                   ;match?
0C74  74 0B                               je    stn_found               ;yes...
0C76  8A 44 04                            mov   al,[si].stn_namel       ;no: skip to next rec
0C79  98                                  cbw
0C7A  03 F0                               add   si,ax
0C7C  83 C6 05                            add   si,stn_rec_fsize
                                        $repeat                         ;and continue

0C81  0C 01             stn_found:      or      al,1                    ;set NZ
0C83  C3                stn_not_found:  ret                             ;return with Z/NZ set
0C84                    stn_lookup      endp


                        ;
                        ; stn_add       Add a station to the end of the list.
                        ;               On input, di points to the new stn_rec.
                        ;               Destroys ax, cx.
                        ;

0C84                    stn_add         proc    near
0C84  57                                push    di
0C85  56                                push    si
0C86  8B F7                             mov     si,di                   ;move from new entry
0C88  8D 3E 0915 R                      lea     di,stn_list-1           ;to zero byte at end of list
0C8C  03 3E 0912 R                      add     di,stn_list_size
0C90  B5 00                             mov     ch,0
0C92  8A 4C 04                          mov     cl,[si].stn_namel       ;length of name
0C95  83 C1 05                          add     cx,stn_rec_fsize;total size of entry to move
```

```
0C98  A1 0912 R                               mov     ax,stn_list_size
0C9B  03 C1                                   add     ax,cx                   ;new size of stn_list
0C9D  3D 0258                                 cmp     ax,max_stn_list
                                              $if     l                       ;not overflowing
0CA2  F6 44 01 80                               test  [si].stn_flags,rumor    ;adding a rumor?
                                                $ifnot z
0CA8  FE 06 0915 R                               inc  rumor_count             ;yes: count it
                                                $endif
0CAC  A3 0912 R                                 mov   stn_list_size,ax
0CAF  F3/ A4                                    rep   movsb                   ;move entire entry
0CB1  C6 05 00                                  mov   byte ptr [di],0         ;put a zero at the end
0CB4  80 0E 0911 R 01                           or    lists_chgd,ch_stn       ;  and flag changed stationlist
0CB9  FE 06 0914 R                              inc   stn_count               ;  and increment count
                                              $endif
0CBD  5E                                      pop     si
0CBE  5F                                      pop     di
0CBF  C3                                      ret
0CC0                          stn_add         endp


                              ;
                              ; stn_remove    Remove a station from the station list.
                              ;
                              ;               On entry, si points to the stn_rec to remove.
                              ;               On exit, si points to the following stn_rec.
                              ;               Destroys: ax.
                              ;
                              ;       Note:   Caller must be aware that dest_stn may be wrong
                              ;               after a station has been removed.
                              ;

0CC0                          stn_remove      proc
0CC0  56                                      push    si
0CC1  57                                      push    di
0CC2  51                                      push    cx
0CC3  F6 44 01 80                             test    [si].stn_flags,rumor    ;removing a rumor?
                                              $ifnot z
0CC9  FE 0E 0915 R                             dec    rumor_count             ;yes: uncount it
                                              $endif
0CCD  8A 44 04                                mov     al,[si].stn_namel       ;length of name
0CD0  04 05                                   add     al,stn_rec_fsize;total length of entry
0CD2  98                                      cbw
0CD3  8B FE                                   mov     di,si                   ;destination to slide to
0CD5  03 F0                                   add     si,ax                   ;source is next entry
0CD7  8B 0E 0912 R                            mov     cx,stn_list_size;compute amount to move
0CDB  81 C1 0916 R                            add     cx,offset cgrp:stn_list
0CDF  2B CE                                   sub     cx,si
0CE1  29 06 0912 R                            sub     stn_list_size,ax;update list size
0CE5  F3/ A4                                  rep     movsb                   ;slide
0CE7  C6 05 00                                mov     byte ptr [di],0         ;put zero at the end
0CEA  80 0E 0911 R 01                         or      lists_chgd,ch_stn       ;flag that stationlist changed
0CEF  FE 0E 0914 R                            dec     stn_count               ;and decrement count
0CF3  59                                      pop     cx
0CF4  5F                                      pop     di
```

```
0CF5  5E                                    pop     si
0CF6  C3                                    ret
0CF7                            stn_remove  endp



                                ;
                                ; group_purge   Look through the group lists and purge any groups which
                                ;               have no current members.
                                ;
                                ;               Destroys:  ax.
                                ;
0CF7                            group_purge     proc    near
0CF7  53                                    push    bx
0CF8  51                                    push    cx
0CF9  52                                    push    dx
0CFA  56                                    push    si
0CFB  57                                    push    di

0CFC  32 FF                                 xor     bh,bh                   ;BH always zero!

                                ;       OR together the group masks of all stations in dx

0CFE  8D 36 0916 R                          lea     si,stn_list             ;start of station list
0D02  33 D2                                 xor     dx,dx                   ;for OR of all group masks
                                            $do
0D04  80 3C 00                                cmp   [si].stn_addr,0
                                              $exitif e
0D09  0B 54 02                                or    dx,[si].stn_groupmask
0D0C  8A 5C 04                                mov   bl,[si].stn_namel
0D0F  8D 70 05                                lea   si,stn_rec_fsize[si+bx]
                                            $repeat

                                ;       Loop through groups, and delete those without a bit on

0D14  8D 36 0B71 R                          lea     si,grp_list
0D18  B8 0001                               mov     ax,1                    ;group counter in ax
                                            $do
0D1B  80 3C 00                                cmp   [si].grp_namel,0
                                              $exitif e
0D20  D1 E2                                   shl   dx,1                    ;look at its bit in dx
                                              $ifnot c                      ;not on: not used!
0D24  80 0E 0911 R 04                           or    lists_chgd,ch_grp       ;flag groups changed
0D29  56                                        push si                       ;save loc of where next will be
0D2A  8B FE                                     mov  di,si                    ;destination to slide to
0D2C  8A 1C                                     mov  bl,[si].grp_namel
0D2E  8D 70 01                                  lea  si,grp_rec_fsize[si+bx] ;source is next grp_record
0D31  8B 0E 0B6E R                              mov  cx,grp_list_size;compute amount to move
0D35  81 C1 0B71 R                              add  cx,offset cgrp:grp_list
0D39  2B CE                                     sub  cx,si
0D3B  29 1E 0B6E R                              sub  grp_list_size,bx;update list size
0D3F  83 2E 0B6E R 01                           sub  grp_list_size,grp_rec_fsize
```

```
0D44  F3/ A4                                    rep  movsb                    ;slide it
0D46  C6 05 00                                  mov  byte ptr [di],0         ;put zero at the end
0D49  FE 0E 0B70 R                              dec  grp_count               ;decrement count
0D4D  38 06 0C4F R                              cmp  dest_grp,al             ;if destination is it or later,
                                                $if  ge
0D53  FE 0E 0C4F R                               dec dest_grp                ;decrement it
                                                $endif
0D57  E8 0D6C R                                 call group_pur_fixstn;fix station groupmasks
0D5A  5E                                        pop  si                      ;restore ptr to slid entry
                                               $else
0D5E  8A 1C                                     mov  bl,[si].grp_namel       ;skip to next entry
0D60  8D 70 01                                  lea  si,grp_rec_fsize[si+bx]
0D63  40                                        inc  ax                      ;count group kept
                                               $endif
                                              $repeat
0D66  5F                                      pop     di                     ;restore regs and return
0D67  5E                                      pop     si
0D68  5A                                      pop     dx
0D69  59                                      pop     cx
0D6A  5B                                      pop     bx
0D6B  C3                                      ret


                              ;       Remove the bit for the removed group from all stations' groupmasks.
                              ;       ax is the group number.  Destroys si, cx, bl.

0D6C                          group_pur_fixstn:
0D6C  8D 36 0916 R                            lea     si,stn_list            ;start of station list
0D70  8B C8                                   mov     cx,ax                  ;save group # in cx
                                              $do
0D72  80 3C 00                                  cmp   [si].stn_addr,0
                                                $exitif e
0D77  8B 44 02                                  mov   ax,[si].stn_groupmask  ;get the mask
0D7A  F8                                        clc                          ;the zero bit to supply
0D7B  51                                        push  cx
0D7C  D3 D0                                       rcl ax,cl                  ;push the bit into C
0D7E  59                                        pop   cx
0D7F  51                                        push  cx
0D80  49                                          dec cx
0D81  D3 C8                                       ror ax,cl                  ;put other bits back
0D83  59                                        pop   cx
0D84  89 44 02                                  mov   [si].stn_groupmask,ax  ;put the mask back
0D87  8A 5C 04                                  mov   bl,[si].stn_namel      ;next station
0D8A  8D 70 05                                  lea   si,stn_rec_fsize[si+bx]
                                              $repeat
0D8F  8B C1                                   mov     ax,cx                  ;restore group # in ax
0D91  C3                                      ret

0D92                          group_purge     endp

                                              assume  ds:nothing,es:nothing
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;
                        ;       UTILITY ROUTINES        Called by any task.
                        ;
                        ;-----------------------------------------------------------------------------
                                %out    -----utilities


                        ;
                        ;     w_print:  Print char in al in the window.  If cr, also prints lf.
                        ;               Destroys nothing.
                        ;

0D92                    w_print         proc    near
0D92  50                                push    ax
0D93  B4 03                             mov     ah,window_print
0D95  E8 0000 E                         call    window
0D98  58                                pop     ax
0D99  C3                                ret
0D9A                    w_print         endp


                        ;
                        ; w_print_dd    Print contents of al (0-99) in decimal.
                        ;               Destroys ax.
                        ;

0D9A                    w_print_dd      proc    near
0D9A  B4 00                             mov     ah,0
0D9C  2E: F6 36 0C9F R                  div     kb10            ;quotient in al, remainder in ah
0DA1  04 30                             add     al,'0'
0DA3  E8 0D92 R                         call    w_print
0DA6  8A C4                             mov     al,ah
0DA8  04 30                             add     al,'0'
0DAA  EB E6                             jmp     short w_print   ;print and return
0DAC                    w_print_dd      endp



                        ;
                        ; w_print_nn    Print contents of al as "nn" in hex.
                        ;               Destroys ax.
                        ;

0DAC                    w_print_nn      proc    near
0DAC  50                                push    ax
0DAD  D0 C8                             ror     al,1
0DAF  D0 C8                             ror     al,1
0DB1  D0 C8                             ror     al,1
0DB3  D0 C8                             ror     al,1
0DB5  E8 0DB9 R                         call    w_print_nibble  ;1st digit
0DB8  58                                pop     ax              ;now 2nd digit
```

```
E8 0DBE R               w_print_nibble: call    to_hex          ;convert to hex
EB D4                                   jmp     w_print         ;print digit and exit
                        w_print_nn      endp


                        ;
                        ; to_hex        convert al's low nibble to hex
                        ;

                        to_hex          proc    near
24 0F                                   and     al,0fh
3C 09                                   cmp     al,9
                                        $if     a
04 07                                     add   al,"A"-"9"-1
                                        $endif
04 30                                   add     al,"0"
C3                                      ret
                        to_hex          endp


                        ;
                        ; do_timestamp  Print a CR in the current window, and then the current
                        ;               time in the form hh:mm if timestamp mode is enabled.
                        ;               Destroys ax.

                        do_timestamp    proc    near
B0 0D                                   mov     al,cr                   ;start with cr
E8 0D92 R                               call    w_print
2E: F6 06 002C R 01                     test    timestamp,1
                                        $ifmot  z
51                                        push  cx
52                                        push  dx
B4 00                                     mov   ah,timer_read           ;get time of day in ticks (18.2/s)
CD 1A                                     int   timer_int               ;in cx:dx
8A C1                                     mov   al,cl                   ;cl=hh because 65536/18.2 = 3600
E8 0D9A R                                 call  w_print_dd              ;print as hh
B0 3A                                     mov   al,':'
E8 0D92 R                                 call  w_print                 ;print :
8B C2                                     mov   ax,dx                   ;get tick minutes in dx:ax
33 D2                                     xor   dx,dx
2E: F7 36 0CA0 R                          div   kw1092                  ;divide by 18.2*60 to get minutes
E8 0D9A R                                 call  w_print_dd              ;print al as mm
B0 20                                     mov   al,' '
E8 0D92 R                                 call  w_print                 ;print blank
5A                                        pop   dx
59                                        pop   cx
                                        $endif
C3                                      ret

                        do_timestamp    endp

                        ;
                        ; int_error     Print internal error message.
                        ;               Input: al=location code
```

```
                        ;                       bx,cx = data to print
                        ;               Destroys ax.
                        ;

49 6E 74 65 72 6E       int_error_msg   db      'Internal error ',0
61 6C 20 65 72 72
6F 72 20 00

                        int_error       proc    near
56                                      push    si
50                                      push    ax
8D 36 0DFA R                            lea     si,int_error_msg
                                        $do                             ;print error msg
2E: AC                                    lods  int_error_msg
3C 00                                     cmp   al,0
                                          $exitif z
E8 0D92 R                                 call  w_print
                                        $repeat
58                                      pop     ax
5E                                      pop     si
E8 0DAC R                               call    w_print_nn              ;print code in al
B0 20                                   mov     al," "
E8 0D92 R                               call    w_print
8A C7                                   mov     al,bh                   ;print bh
E8 0DAC R                               call    w_print_nn
8A C3                                   mov     al,bl                   ;print bl
E8 0DAC R                               call    w_print_nn
B0 20                                   mov     al," "
E8 0D92 R                               call    w_print
8A C5                                   mov     al,ch                   ;print ch
E8 0DAC R                               call    w_print_nn
8A C1                                   mov     al,cl                   ;print cl
E8 0DAC R                               call    w_print_nn
B0 0D                                   mov     al,cr
E8 0D92 R                               call    w_print
C3                                      ret
                        int_error       endp


                        ;
                        ; release_rbs   Release any rbs allocated.
                        ;               Restore the L4_debug flag.
                        ;

                        release_rbs     proc    near
2E: C5 3E 0C66 R                        lds     di,l4_publics           ;restore l4_debug
2E: A1 0C6A R                           mov     ax,our_l4_debug
89 45 06                                mov     ds:[di].l4_debug,ax
2E: C5 36 0C5E R                        lds     si,rcv_rb               ;release receiver's rb
E8 0E5D R                               call    release_an_rb
2E: C5 36 0C62 R                        lds     si,snd_rb               ;release sender's rb

8C D8                   release_an_rb:  mov     ax,ds
```

```
0B C0                                   or      ax,ax                   ;is it null?
                                        $ifnot  z
                                        14_call release_rb              ;no: release it
                                        $endif
C3                                      ret
                        release_rbs     endp
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;
                        ; DISPATCH      Dispatch the next task, in round-robin fashion.
                        ;               NO registers are destroyed during a task switch, except flags.
                        ;               CS and SS registers are assumed constant for all tasks.
                        ;
                        ;-----------------------------------------------------------------------------

                        dispatch        proc    near

                        ;       Deactivate current task

50                              push    ax                      ;save everything
53                              push    bx
51                              push    cx                      ;WARNING: init_task knows how
52                              push    dx                      ;  much we save!
55                              push    bp
56                              push    si
57                              push    di
1E                              push    ds
06                              push    es
2E: 8B 1E 000F R                mov     bx,curtask              ;get current task block pointer
2E: 89 A7 0007 R                mov     tcbs[bx],sp             ;save stack pointer there

83 C3 02                        add     bx,tcbsize              ;change to offset of next tcb
83 FB 08                        cmp     bx,ntasks*tcbsize
                                $if     ae
BB 0000                           mov   bx,0                    ;wraparound to task 0
                                $endif
2E: 89 1E 000F R                mov     curtask,bx

                        ;       Activate new task

2E: 8B 1E 000F R                mov     bx,curtask
2E: 8B A7 0007 R                mov     sp,tcbs[bx]             ;get stack pointer
07                              pop     es                      ;restore everything
1F                              pop     ds
5F                              pop     di
5E                              pop     si
5D                              pop     bp
5A                              pop     dx
59                              pop     cx
5B                              pop     bx
58                              pop     ax
C3                              ret                             ;resume that task

                        dispatch        endp
```

```
                                page

                        ;-----------------------------------------------------------------------------
                        ;
                        ; dispatch_start       This is the keyboard/timer/l4_exit interrupt task,
                        ;                      and runs with the registers of the interrupt routine.
                        ;                      It's only function is to start and stop the
                        ;                      task dispatcher so that the other tasks run.
                        ;
                        ; No registers can be destroyed!  Also, use as little original stack as possible.
                        ;
                        ; The caller should already have ensured that L4 is not busy.
                        ;
                        ;-----------------------------------------------------------------------------

                        dispatch_start  proc    near
50                              push    ax              ;save the only reg we use

FA                              cli
2E: F6 06 001B R 01             test    tasks_active,1  ;last-chance recursion check
                        $if     z
2E: C6 06 001B R 01             mov     tasks_active,1  ;signal that tasks are active
2E: 8C 16 0011 R                mov     savss,ss        ;save caller's ss:sp
2E: 89 26 0013 R                mov     savsp,sp
8C C8                           mov     ax,cs           ;setup our own stack
8E D0                           mov     ss,ax
8D 26 0F00 R                    lea     sp,stack0       ;as task zero
2E: C7 06 000F R 0000           mov     curtask,0
=B                              sti

=8 0EF3 R                       call    busy_rb_check           ;check for other open connections

                                $do
=8 0E69 R                         call  dispatch                ;run other tasks until
2E: F6 06 0024 R 01               test  do_stop,1               ;told to stop
                                $repeatwhile z
2E: C6 06 0024 R 00             mov     do_stop,0

=A                              cli                     ;recover caller's stack
)0                              nop
2E: 8E 16 0011 R                mov     ss,cs:savss
2E: 8B 26 0013 R                mov     sp,cs:savsp
2E: C6 06 001B R 00             mov     tasks_active,0  ;tasks are not active

                        $endif                          ;recursion check
=B                              sti

)8                              pop     ax              ;restore reg
=3                              ret                     ;return to kb interrupt rtn

                        dispatch_start  endp
```

```
                                page


                        ;-----------------------------------------------------------------------------
                        ;
                        ; busy_rb_check         Check if any rb's in the system have open connections.
                        ;                       Set the "other_conn_busy" flag accordingly.
                        ;                       Destroys no registers.
                        ;
                        ; This flag is used by the tasks to avoid time-consuming operations if
                        ; there are other processes in the machine that need to service the network
                        ; and might timeout otherwise, such as an open connection to the fileserver.
                        ;
                        ; The operations currently avoided when other connection are busy are:
                        ;
                        ;       1. Opening the windows to allow user interaction.
                        ;       2. Sending probe messages to verify rumors.
                        ;
                        ; Note that this technique will prevent the windows from ever being opened
                        ; if a long-duration connection is maintained, such as for ms/net!
                        ;
                        ;-----------------------------------------------------------------------------

                        busy_rb_check   proc    near
50                                      push    ax
1E                                      push    ds
56                                      push    si
2E: C6 06 0021 R 01                     mov     other_conn_busy,1       ;assume we will find someone
2E: C5 36 0C66 R                        lds     si,14_publics
C5 74 14                                lds     si,dword ptr ds:[si].active_head ;head of rb chain
                                        $do
8C D8                                     mov   ax,ds                   ;is ptr zero?
0B C0                                     or    ax,ax
                                          $exitif z                     ;yes: end of chain - no action
8A 44 08                                  mov   al,ds:[si].conn_status
3C 00                                     cmp   al,cn_established       ;check for open connection
74 0E                                     je    busy_rb_xit
3C 05                                     cmp   al,cn_accept_wait       ;or about-to-be-open connection
74 0A                                     je    busy_rb_xit
C5 34                                     lds   si,dword ptr ds:[si].next ;next rb in the chain
                                        $repeat
2E: C6 06 0021 R 00                     mov     other_conn_busy,0       ;nothing busy

5E                      busy_rb_xit:    pop     si
1F                                      pop     ds
58                                      pop     ax
C3                                      ret

                        busy_rb_check   endp
```

```
                        page
                        %out    -----interrupt routines

                ;-----------------------------------------------------------------------------
                ;
                ;  Keyboard read interrupt (int 16h) intercept routine.
                ;
                ;       Start the TALK task dispatcher if a message is incoming, or
                ;       if the trigger character has been typed.
                ;
                ;       Enter with the keyboard interrupt request code in ah.
                ;       This routine MUST MODIFY NO REGISTERS EXCEPT AX, and must
                ;       use as little stack as possible.
                ;
                ;       This routine guards against recursion, and won't do anything if the
                ;       tasks dispatcher is already running, or if L4 is busy.
                ;
                ;       This supports the extended function code "key_talk", which returns
                ;       NOT key_talk in AH to indicate we are present, and status bits
                ;       in AL to indicate whether the window are open on the screen.
                ;       This is used by NWCMDS to know whether to open its window.
                ;
                ;       This also supports the extended function code "key_talk_send"
                ;       to send messages to another station's TALK window.
                ;
                ;-----------------------------------------------------------------------------

                key_int_rtn     proc    far
FA                              cli                     ;insure disable if called by other trap

80 FC 63                        cmp     ah,key_talk         ;special nestar extended function
                                $if     e                   ;  for "talk info"?
E9 102A R                         jmp   do_key_talk         ;yes: do it whether active or not
                                $endif

80 FC 60                        cmp     ah,key_talk_stop    ;special nestar extended function
                                $if     e                   ; for "talk suspend"?
E9 106E R                         jmp   do_key_talk_stop    ;yes: do it whether active or not
                                $endif

80 FC 5F                        cmp     ah,key_talk_start   ;special nestar extended function
                                $if     e                   ; for "talk resume"?
E9 1078 R                         jmp   do_key_talk_start   ;yes: do it whether active or not
                                $endif


2E: F6 06 001B R 01             test    tasks_active,1          ;tasks running?
75 1F                           jnz     kb_int_next             ;yes: don't do anything

2E: F6 06 0028 R 01             test    suspended,1             ;have we been stopped?
75 17                           jnz     kb_int_next             ;yes: don't do anything
```

```
2E: F6 06 0020 R 01                     test    kb_int_busy,1           ;recursion test
75 0F                                   jnz     kb_int_next             ; (actually unnecessary)

1E                                      push    ds
56                                      push    si
2E: C5 36 0C66 R                        lds     si,14_publics           ;get ptr to 14 publics
F6 44 08 01                             test    ds:[si].14_in_use,1     ;is 14 busy?
5E                                      pop     si
1F                                      pop     ds
                                        $ifnot  z
2E: FF 2E 0C75 R        kb_int_next:      jmp   old_key_vector          ;to to next kb routine
                                        $endif


                        ;       Do one-time-only dispatch to initialize all tasks and allow
                        ;       the broadcast message to be sent by the send_task.

2E: F6 06 0025 R 01                     test    first_dispatch,1;has it been done?
                                        $ifnot  z                       ;no
E8 0EF3 R                                 call  busy_rb_check           ;check for busy fileserver connection
2E: F6 06 0021 R 01                       test  other_conn_busy,1       ; (avoids 30 second retry delay!)
75 E8                                     jnz   kb_int_next             ;busy: can't initialize now
2E: C6 06 0025 R 00                       mov   first_dispatch,0;clear the flag
E8 0EA0 R                                 call  dispatch_start          ;and do it
                                        $endif


                        ;       Do a dispatch if a request to open the windows is queued,
                        ;       or if there is an incoming message.

2E: F6 06 0023 R 01                     test    do_open,1
                                        $ifnot  z
E8 0EA0 R                                 call  dispatch_start
                                        $endif

E: F6 06 0026 R 01                      test    open_recvd,1
                                        $ifnot  z
E8 0EA0 R                                 call  dispatch_start
                                        $endif


                        ;       Determine keyboard interrupt request type

80 FC 00                                cmp     ah,key_read             ;read character request?
74 0D                                   je      do_key_read
80 FC 01                                cmp     ah,key_stat             ;read status request?
74 40                                   je      do_key_stat
80 FC 61                                cmp     ah,key_talk_send;nestar extended cmd to send_msg?
75 BA                                   jne     kb_int_next             ;no: pass it on
E9 103D R                               jmp     do_key_send

                        ;       read key

FC                      do_key_read:    pushf                           ;get the character requested
```

```
B4 00                                   mov     ah,key_read             ;
FA                                      cli                             ;   (enter him disabled)
2E: FF 1E 0C75 R                        call    old_key_vector          ;   from the next kb routine
2E: 3A 06 005D R                        cmp     al,keychar              ;our trigger character?
75 6B                                     jne   key_int_exit            ;no: exit to caller
0A C0                                   or      al,al                   ;check for extended scan code
                                        $if     z
2E: 3A 26 005E R                          cmp   ah,keychar_scan
75 60                                     jne   key_int_exit
                                        $endif
2E: C6 06 0023 R 01                     mov     do_open,1               ;open windows
2E: C6 06 0029 R 00                     mov     popped_up,0
2E: C6 06 0020 R 01                     mov     kb_int_busy,1
E8 0EA0 R                               call    dispatch_start          ;and process until closed
2E: C6 06 0020 R 00                     mov     kb_int_busy,0
EB C8                                   jmp     short do_key_read       ;then read another key for caller

                        ;       read status

9C                      do_key_stat:    pushf                           ;get status
B4 01                                   mov     ah,key_stat             ;
FA                                      cli                             ;   (enter him disabled)
2E: FF 1E 0C75 R                        call    old_key_vector          ;   from the next kb routine
74 38                                     jz    key_int_exit            ;no key: exit with Z set
2E: 3A 06 005D R                        cmp     al,keychar              ;our trigger character?
75 31                                     jne   key_int_exit            ;no: exit with NZ set
0A C0                                   or      al,al                   ;check for extended scan code
                                        $if     z
2E: 3A 26 005E R                          cmp   ah,keychar_scan
75 26                                     jne   key_int_exit            ;no: exit with NZ set
                                        $endif
B4 00                                   mov     ah,key_read             ;consume the key
9C                                      pushf
FA                                      cli                             ; (enter him disabled)
2E: FF 1E 0C75 R                        call    old_key_vector
2E: C6 06 0023 R 01                     mov     do_open,1               ;open windows
2E: C6 06 0029 R 00                     mov     popped_up,0
2E: C6 06 0020 R 01                     mov     kb_int_busy,1
E8 0EA0 R                               call    dispatch_start          ;and process until closed
2E: C6 06 0020 R 00                     mov     kb_int_busy,0
EB BD                                   jmp     short do_key_stat       ;then read status again for caller

CA 0002                 key_int_exit:   ret     2                       ;return to caller; return new flags


                        ;
                        ;               Nestar extended int 16h functions
                        ;


                        ;
                        ;       Return TALK extended info
```

```
                        ;
                        ;       ah=key_talk
                        ;       return ah=not key_talk, status bits ("key_ext_xxx") in al

 4 9C                   do_key_talk:    mov     ah,not key_talk         ;return complemented fct code
 E: F6 06 0022 R 01                     test    windows_open,1          ; to indicate we are present
                                        $if     z
 0 02                                     mov   al,key_ext_idle         ;windows not on screen
                                        $else
 0 01                                     mov   al,key_ext_active       ;windows are on screen
                                        $endif
 B EA                                   jmp     short key_int_exit


                        ;
                        ;       Send a TALK message.
                        ;
                        ;       ah=key_talk_send;  al=stn addr, ds:si=message ending with CR
                        ;       return ah=not key_talk_send; al=status

 E: A2 005B R           do_key_send:    mov     send_msg_stn,al         ;save stn addr
 E: 89 36 0057 R                        mov     word ptr send_msg_ptr,si;save message ptr
 E: 8C 1E 0059 R                        mov     word ptr send_msg_ptr+2,ds
 E: C6 06 005C R 00                     mov     send_msg_stat,0         ;clear status
 E: C6 06 0027 R 01                     mov     send_msg_rqst,1         ;turn on request
 E: C6 06 0020 R 01                     mov     kb_int_busy,1
 8 0EA0 R                               call    dispatch_start          ;start tasks to process
 E: C6 06 0020 R 00                     mov     kb_int_busy,0
 4 9E                                   mov     ah,not key_talk_send    ;return complemented fct code
 E: A0 005C R                           mov     al,send_msg_stat;and send status
 B B9                                   jmp     short key_int_exit

                        ;
                        ;       Suspend TALK.
                        ;
                        ;       ah=key_talk_stop;  al=0 (reserved for future use)
                        ;       Return ah=not key_talk_stop

                        do_key_talk_stop:
 E: C6 06 0028 R 01                     mov     suspended,1             ;(pbs if active now?)
 4 9F                                   mov     ah,not key_talk_stop    ;return complemented fct code
 3 AF                                   jmp     short key_int_exit


                        ;
                        ;       Resume TALK.
                        ;
                        ;       ah=key_talk_start;  al=0 (reserved for future use)
                        ;       Return ah=not key_talk_start

                        do_key_talk_start:
 : C6 06 0028 R 00                      mov     suspended,0             ;unsuspend
 A0                                     mov     ah,not key_talk_start   ;return complemented fct code
 A5                                     jmp     short key_int_exit
```

```
key_int_rtn     endp
```

```
                        page

                    ;-----------------------------------------------------------------------
                    ;
                    ;       Check for incoming messages if we haven't checked for a while.
                    ;       If a message is arriving, startup the task dispatcher to process it.
                    ;
                    ;       This is called from interrupt routines, so should conserve stack space.
                    ;       Destroys ax and flags.
                    ;
                    ;-----------------------------------------------------------------------

                    check_incoming  proc    near
2E: F6 06 0026 R 01                 test    open_recvd,1            ;is msg incoming already?
                        $if z                                       ;yes: don't bother to check L4
51                                  push    cx
52                                  push    dx
B4 00                               mov     ah,timer_read           ;get current time
CD 1A                               int     timer_int               ;
8A C2                               mov     al,dl                   ;low byte of time
2E: 2A 06 0C5B R                    sub     al,last_open_rcv;now-then (even if wraparound!)
3C 03                               cmp     al,how_often            ;long enough ago?
5A                                  pop     dx                      ;
59                                  pop     cx

                            $if     ae                              ;was long enough ago
2E: 88 16 0C5B R                    mov     last_open_rcv,dl;save time now

1E                                  push    ds
56                                  push    si
2E: C5 36 0C66 R                    lds     si,l4_publics           ;get ptr to l4 publics
F6 44 08 01                         test    ds:[si].l4_in_use,1     ;is l4 busy?
                                $if z                               ;no: can check
FA                                  cli
2E: 8C 16 0011 R                    mov     savss,ss                ;save caller's ss:sp
2E: 89 26 0013 R                    mov     savsp,sp
8C C8                               mov     ax,cs                   ;setup our own stack
8E D0                               mov     ss,ax
8D 26 0F00 R                        lea     sp,stack0               ;borrow task 0's
FB                                  sti
2E: C5 36 0C5E R                    lds     si,rcv_rb               ;get ptr to rb
2E: A1 0C5C R                       mov     ax,wks
89 44 30                            mov     [si].hdr_src_socket,ax  ;listen socket
                                    l4_call open_recv
                                    $if     c                       ;message is coming in
2E: C6 06 0026 R 01                   mov   open_recvd,1            ;  flag to that effect
                                    $endif
FA                                  cli                             ;recover caller's stack
90                                  nop
2E: 8E 16 0011 R                    mov     ss,cs:savss
2E: 8B 26 0013 R                    mov     sp,cs:savsp
FB                                  sti
                                $endif                              ;l4 busy check
5E                                  pop     si
```

```
F                                       pop     ds

                                $endif                                  ;long-ago check

                            $endif                                      ;already incoming check


E: F6 06 0026 R 01                      test    open_recvd,1            ;is a msg incoming?
                                        $ifnot  z                       ;yes,
E                                         push  ds
6                                         push  si
E: C5 36 0C66 R                           lds   si,l4_publics           ;get ptr to l4 publics
6 44 08 01                                test  ds:[si].l4_in_use,1     ;is l4 busy?
E                                         pop   si
F                                         pop   ds
                                          $if   z
8 0EA0 R                                   call dispatch_start          ;no: start the tasks
                                          $endif                        ;l4 busy check
                                        $endif


3                                       ret
                        check_incoming  endp
```

```
                        page

                ;-----------------------------------------------------------------------
                ;
                ;       L4 exit "interrupt" routine
                ;
                ;       This is called by the transport level network routines after the
                ;       l4_in_use flag has been cleared.  We use it as a way to get
                ;       "asynchronous" control at a time when it is safe to call L4.
                ;
                ;
                ;-----------------------------------------------------------------------

                l4_exit_rtn     proc    far
C                               pushf                           ;save flags, ax
D                               push    ax
A                               cli
E: A0 001F R                    mov     al,l4_exit_busy         ;recursion and busy checks
E: 0A 06 001B R                 or      al,tasks_active
E: 0A 06 001E R                 or      al,timer_int_busy
E: 0A 06 0020 R                 or      al,kb_int_busy
E: 0A 06 0028 R                 or      al,suspended
                                $if     z
E: C6 06 001F R 01                mov   l4_exit_busy,1
3 1082 R                          call  check_incoming          ;check for incoming messages
E: C6 06 001F R 00                mov   l4_exit_busy,0
                                $endif
3                               pop     ax
)                               popf                            ;restore flags, ax
3                               sti
E: FF 2E 0C6C R                 jmp     old_l4_exit             ;chain to previous l4 exit rtn
                l4_exit_rtn     endp
```

```
                        page

                ;-----------------------------------------------------------------------
                ;
                ;
                ;               Timer interrupt routine
                ;
                ;
                ;       1. Handle noise (tweedle) generation.
                ;       2. Count ticks
                ;       3. Check for incoming messages
                ;
                ; We used to hang off the software timer vector (int 1ch), but that doesn't
                ; work because the interrupt controller had not yet been reset, causing the
                ; clock not to run, which causes L4 to hang waiting for packets to time out.
                ; We now hang off the hardware interrupt and execute the existing timer
                ; interrupt routine to completion before starting any network activity.
                ; The recursion flag prevents subsequent clock ticks from causing problems.
                ; Note that the recursion flag is set late so tweedles can be processed even
                ; while the tasks are running after having been started by this interrupt rtn.
                ;
                ;-----------------------------------------------------------------------

                timer_int_rtn   proc    far
9C                              pushf                           ;push flags to simulate hw int
2E: FF 1E 0C79 R                call    old_time_vec            ;execute the hw interrupt rtn
50                              push    ax                      ;save regs
53                              push    bx

2E: FF 06 0C7D R                inc     ticks                   ;count ticks

                ;       Do tweedling

2E: 8B 1E 0C7F R                mov     bx,tweedle_ptr          ;get ptr to sound array
83 FB 00                        cmp     bx,0
                        $ifnot  e                               ;if sound in progress,
2E: 8A 27                       mov     ah,cs:[bx]              ;get sound array value
0A E4                           or      ah,ah

                                $ifnot  e                       ;more sounds to make
B0 A6                           mov     al,timer_ctrl_msb       ;setup to load msb
E6 43                           out     timer_ctrl,al
8A C4                           mov     al,ah                   ;load msb from array value
E6 42                           out     timer_ch2,al
E4 61                           in      al,kb_ctrl
0C 03                           or      al,kb_ctrl_spkr         ;enable speaker
E6 61                           out     kb_ctrl,al
43                              inc     bx                      ;increment ptr to array
2E: 89 1E 0C7F R                mov     tweedle_ptr,bx

                                $else                           ;stop the sounds
E4 61                           in      al,kb_ctrl
24 FC                           and     al,255-kb_ctrl_spkr     ;turn off the speaker
E6 61                           out     kb_ctrl,al
```

```
2E: C7 06 0C7F R 0000                   mov     tweedle_ptr,0           ;zero ptr to sound array
                                        $endif

                                $endif                                  ;sound in progress


                        ;       Do incoming message check, if things are otherwise quiescent.
                        ;       (Only the keyboard interrupt is allowed to be active.)

2E: A0 001E R                           mov     al,timer_int_busy       ;recursion check
2E: 0A 06 001B R                        or      al,tasks_active         ;other activity checks
2E: 0A 06 001F R                        or      al,14_exit_busy
2E: 0A 06 0028 R                        or      al,suspended
                                        $if     z
2E: C6 06 001E R 01                       mov   timer_int_busy,1;flag timer interrupt busy
E8 1082 R                                 call  check_incoming          ;check for incoming messages
2E: C6 06 001E R 00                       mov   timer_int_busy,0;clear timer interrupt busy flag
                                        $endif                          ;recursion check

5B                                      pop     bx                      ;restore regs
58                                      pop     ax
CF                                      iret                            ;return from interrupt
                        timer_int_rtn   endp
```

```
                                page


                        ;
                        ;       talk_exit:      Shutdown everything
                        ;
                        talk_exit       proc    near

                        ;       Remove the timer tick interrupt routine

A                               cli
E 0000                          mov     si,0            ;point es:si at interrupt vector
E C6                            mov     es,si
E 0020                          mov     si,timer_tick*4
E: A1 0C79 R                    mov     ax,word ptr old_time_vec ;bx:ax is old pointer
E: 8B 1E 0C7B R                 mov     bx,word ptr old_time_vec+2
6: 89 04                        mov     es:[si],ax
6: 89 5C 02                     mov     es:[si+2],bx
B                               sti

                        ;       Remove the keyboard interrupt routine

E 0058                          mov     si,key_int*4
E: A1 0C75 R                    mov     ax,word ptr old_key_vector
E: 8B 1E 0C77 R                 mov     bx,word ptr old_key_vector+2
6: 89 04                        mov     es:[si],ax
6: 89 5C 02                     mov     es:[si+2],bx

                        ;       Release network resources

E: A1 0C5C R                    mov     ax,wks                  ;cancel the listen
                                14_call ignore
8 0E44 R                        call    release_rbs             ;release the rbs

3                               ret
                        talk_exit       endp

                        cseg            ends
                                        end
```

Name	Length
. . . . . .000E	
. 0001	
. 0001	
. 0001	
. 0002	
. 0002	
. 0006	
. 0008	
T 0008	
. 0009	
. 0004	
. 0001	
. 0001	
. 0006	
. 0006	
. 0001	
. 0001	
. 0001	
. 0001	
. 0007	
OP. 0007	
TIL 0007	
ILE 0007	
. 0003	
. 0002	

s and records:

Name	Width / Shift	# fields / Width	Mask	Initial
DER 002E	0015			
E	0000			
. . . . .	0001			
NUM	0002			
T	0003			
M	0004			
H	0006			
TRL	0008			
TYPE.	0009			
TWORK	000A			
ST.	000E			
CKET.	0014			
WORK.	0016			
T	001A			
KET	0020			
RL.	0022			
PE.	0023			
ID.	0024			
. . . . .	0026			
. . . . .	0028			

```
                                  002A
UM. . . . . .                    002C
. . . . . .0003                   0002
EL. . . . . .                     0000
E . . . . . .                     0001
. . . . . .0006                   0001
UBS . . . . . .004B               001D
ION . . . . . .                   0000
URES. . . . . .                   0002
US. . . . . .                     0004
. . . . . .                       0005
G . . . . . .                     0006
SE. . . . . .                     0008
T_MODE. . . . . .                 0009
ER. . . . . .                     000A
AD. . . . . .                     0010
HEAD. . . . . .                   0014
CEPT_WAIT . . . . . .             0018
TRIES . . . . . .                 001A
K_WAIT. . . . . .                 001C
GE_TRIES. . . . . .               001E
T_WAIT. . . . . .                 0020
KEEP. . . . . .                   0022
AIT . . . . . .                   0024
KIO_VECTOR. . . . . .             0026
ECTOR . . . . . .                 002A
IT_VECTOR . . . . . .             002E
. . . . . .                       0032
RE1 . . . . . .                   0034
RE2 . . . . . .                   0036
RE3 . . . . . .                   0038
RE4 . . . . . .                   003A
T_VECTOR. . . . . .               003C
ST. . . . . .                     0040
. . . . . .                       0046
FROM_NET. . . . . .               004A
. . . . . .0086                   0043
. . . . . .                       0000
. . . . . .                       0004
_MODE. . . . . .                  0006
SE. . . . . .                     0007
ATUS. . . . . .                   0008
ATUS. . . . . .                   0009
. . . . . .                       000A
NGTH. . . . . .                   000E
_CODE . . . . . .                 0010
AGE. . . . . .                    0011
ET_NUM . . . . . .                0012
MENT . . . . . .                  0013
KSUM . . . . . .                  0014
NGTH . . . . . .                  0016
S_CTRL . . . . . .                0018
ET_TYPE. . . . . .                0019
_NETWORK . . . . . .              001A
```

```
EST_HOST. . . . . .  001E
EST_SOCKET. . . . . .  0024
RC_NETWORK. . . . . .  0026
RC_HOST . . . . . .  002A
RC_SOCKET . . . . . .  0030
ONN_CTRL. . . . . .  0032
ATA_TYPE. . . . . .  0033
OURCE_ID. . . . . .  0034
EST_ID. . . . . .  0036
EQ_NUM. . . . . .  0038
CK_NUM. . . . . .  003A
LOC_NUM. . . . . .  003C
STATUS. . . . . .  003E
YPE. . . . . .  003F
TR . . . . . .  0040
IMIT . . . . . .  0044
ENGTH. . . . . .  0046
ACCEPT_WAIT. . . . . .  0048
N_TRIES. . . . . .  004A
ACK_WAIT . . . . . .  004C
SAGE_TRIES . . . . . .  004E
PKT_WAIT . . . . . .  0050
ALID . . . . . .  0052
YPE. . . . . .  0053
C. . . . . .  0054
ST . . . . . .  0055
RE1. . . . . .  0056
RE2. . . . . .  0058
RE3. . . . . .  005A
RE4. . . . . .  005C
TATE . . . . . .  005E
TATE . . . . . .  005F
HANGED . . . . . .  0060
URSOR. . . . . .  0062
EMAINING . . . . . .  0066
K_REQ. . . . . .  0068
Q_VALID. . . . . .  006A
TATE . . . . . .  006B
HANGED . . . . . .  006C
URSOR. . . . . .  006E
Q. . . . . .  0072
K. . . . . .  0074
LOC. . . . . .  0076
UF . . . . . .  0078
OR . . . . . .  007A
TART . . . . . .  007E
ETRIES . . . . . .  0080
TART_SEQ . . . . . .  0082
AGS. . . . . .  0084
. . . . . .  0085
. . . . . .0007  0005
OR . . . . . .  0000
AGS. . . . . .  0001
OUPMASK. . . . . .  0002
```

```
EL. . . . . .  0004
E . . . . . .  0005
REC . . . . . .000A  0005
PT_WAIT . . . . . .  0000
IES . . . . . .  0002
WAIT. . . . . .  0004
_TRIES. . . . . .  0006
WAIT. . . . . .  0008
```

and Groups:

N a m e	Size	Align	Combine	Class
. . . . . .GROUP				
. . . . .	0000	PARA	COMMON	'HSEGCL'
. . . . .	3953	PARA	COMMON	'VSEGCL'
. . . . .	11DD	PARA	PUBLIC	'CODE'
NT. 100E	PARA	NONE		

N a m e	Type	Value	Attr
. . . . . .Number	FFFF		
. . . . . .N PROC	0038	CSEG	Global Length =0027
. . . . . .Alias	C_TRUE		
OWN . . . . . .L BYTE	002D	VSEG	
. . . . . .L BYTE	002B	VSEG	
. . . . . .Number	0002		
. . . . . .Number	0004		
. . . . . .Number	0003		
. . . . . .Number	0004		
. . . . . .Number	0007		
. . . . . .L NEAR	008C	CSEG	
CNET. . . . . .L BYTE	0C58	VSEG	
. . . . . .Number	0008		
. . . . . .L NEAR	009C	CSEG	
.E. . . . . .N PROC	0091	CSEG	Length =0024
HECK. . . . . .N PROC	0EF3	CSEG	Length =0030
T. . . . . .L NEAR	0F1F	CSEG	
. . . . . .L BYTE	0030	VSEG	
. . . . . .L BYTE	0033	VSEG	
.INE. . . . . .L BYTE	0035	VSEG	
. . . . . .L BYTE	0034	VSEG	
.L. . . . . .L BYTE	0032	VSEG	
MING . . . . . .N PROC	1082	CSEG	Length =0084
ANGES. . . . . .N PROC	06BE	CSEG	Length =003F
_XIT . . . . . .L NEAR	06FC	CSEG	
. . . . . .Number	0004		
. . . . . .Number	0008		
. . . . . .Number	0001		
. . . . . .Number	0002		
WAIT . . . . . .Number	0005		
SHED . . . . . .Number	0000		

```
. . . . . .Number  0002
ONN. . . . . .Number  0001
RCVD . . . . . .Number  0006
ERROR. . . . . .Number  0004
_ERROR . . . . . .Number  0003
 . . . . . .L BYTE  0031  VSEG
T. . . . . .L NEAR  0C11  CSEG
TS . . . . . .N PROC  0B36  CSEG  Length =0038
 . . . . . .Number  000D
 . . . . . .L WORD  000F  VSEG
 . . . . . .Number  0000
 . . . . . .Number  FFFF
 . . . . . .L BYTE  0C4F  VSEG
 . . . . . .L BYTE  0C50  VSEG
 . . . . . .Number  0013
 . . . . . .N PROC  0E69  CSEG  Length =0037
_START . . . . . .N PROC  0EA0  CSEG  Length =0053
KIT. . . . . .Number  004C
KITSTAY. . . . . .Number  0031
RINTC. . . . . .Number  0002
RINTS. . . . . .Number  0009
 . . . . . .Number  0021
 . . . . . .L NEAR  034B  CSEG
EAD. . . . . .L NEAR  0FAC  CSEG
END. . . . . .L NEAR  103D  CSEG
TAT. . . . . .L NEAR  0FE4  CSEG
ALK. . . . . .L NEAR  102A  CSEG
ALK_START. . . . . .L NEAR  1078  CSEG
ALK_STOP . . . . . .L NEAR  106E  CSEG
 . . . . . .L BYTE  0023  VSEG
P. . . . . .L NEAR  02BA  CSEG
 . . . . . .L BYTE  0024  VSEG
TAMP . . . . . .N PROC  0DC9  CSEG  Length =0031
TOM_ROW. . . . . .L BYTE  0048  VSEG
T_COL. . . . . .L BYTE  0045  VSEG
ERRIGHT. . . . . .L WORD  0047  VSEG
HT_COL . . . . . .L BYTE  0047  VSEG
_ROW . . . . . .L BYTE  0046  VSEG
ERLEFT . . . . . .L WORD  0045  VSEG
 . . . . . .Number  0011
 . . . . . .Number  001B
 . . . . . .Number  0000
P. . . . . .L NEAR  0062  CSEG
 . . . . . .N PROC  005F  CSEG  Length =0032
 EXIT. . . . . .L NEAR  0090  CSEG
SPATCH . . . . . .L BYTE  0025  VSEG
SIZE . . . . . .Number  0014
 TYPE. . . . . .Number  0099
 REC . . . . . .N PROC  0B6E  CSEG  Length =003B
AR. . . . . .L NEAR  015D  CSEG
SIG. . . . . .Number  6272
 . . . . . .L NEAR  030B  CSEG
S . . . . . .L WORD  069C  CSEG
GE. . . . . .N PROC  0CF7  CSEG  Length =009B
```

```
_FIXSTN . . . . . .L NEAR  0D6C    CSEG
OW . . . . . .L BYTE  160F    VSEG
 . . . . . .L BYTE  0B70    VSEG
 . . . . . .L BYTE  0B71    VSEG
IZE. . . . . .L WORD  0B6E    VSEG
_MAP . . . . . .L BYTE  0C3E    VSEG    Length =0011
IZE. . . . . .Number  0001
E. . . . . .Number  00BB
M_ROW. . . . . .L BYTE  0040    VSEG
COL. . . . . .L BYTE  003D    VSEG
RIGHT. . . . . .L WORD  003F    VSEG
_COL . . . . . .L BYTE  003F    VSEG
OW . . . . . .L BYTE  003E    VSEG
LEFT . . . . . .L WORD  003D    VSEG
 . . . . . .Number  0003
 . . . . . .L WORD  1000    NIC_SEGMENT
 . . . . . .L NEAR  00B5    CSEG
 . . . . . .L NEAR  0207    CSEG
 . . . . . .L NEAR  0243    CSEG
 . . . . . .L NEAR  0240    CSEG
 . . . . . .L NEAR  022B    CSEG
 . . . . . .L NEAR  0220    CSEG
 . . . . . .L NEAR  0238    CSEG
 . . . . . .L NEAR  0279    CSEG
 . . . . . .L NEAR  0276    CSEG
 . . . . . .L NEAR  0264    CSEG
 . . . . . .L NEAR  0259    CSEG
 . . . . . .L NEAR  026E    CSEG
 . . . . . .L NEAR  0290    CSEG
 . . . . . .L NEAR  0290    CSEG
 . . . . . .L NEAR  028D    CSEG
 . . . . . .L NEAR  00D6    CSEG
 . . . . . .L NEAR  02B3    CSEG
 . . . . . .L NEAR  02B0    CSEG
 . . . . . .L NEAR  02A5    CSEG
 . . . . . .L NEAR  02D0    CSEG
 . . . . . .L NEAR  02D9    CSEG
 . . . . . .L NEAR  02E2    CSEG
 . . . . . .L NEAR  02EB    CSEG
 . . . . . .L NEAR  02F2    CSEG
 . . . . . .L NEAR  0308    CSEG
 . . . . . .L NEAR  0307    CSEG
 . . . . . .L NEAR  0348    CSEG
 . . . . . .L NEAR  00D6    CSEG
 . . . . . .L NEAR  033E    CSEG
 . . . . . .L NEAR  0351    CSEG
 . . . . . .L NEAR  0356    CSEG
 . . . . . .L NEAR  03A2    CSEG
 . . . . . .L NEAR  03B4    CSEG
 . . . . . .L NEAR  03CB    CSEG
 . . . . . .L NEAR  03D2    CSEG
 . . . . . .L NEAR  03E6    CSEG
 . . . . . .L NEAR  042D    CSEG
 . . . . . .L NEAR  03FF    CSEG
```

```
. . . . . .L NEAR  0418  CSEG
. . . . . .L NEAR  00EF  CSEG
. . . . . .L NEAR  0418  CSEG
. . . . . .L NEAR  0418  CSEG
. . . . . .L NEAR  0450  CSEG
. . . . . .L NEAR  044B  CSEG
. . . . . .L NEAR  0467  CSEG
. . . . . .L NEAR  046E  CSEG
. . . . . .L NEAR  0482  CSEG
. . . . . .L NEAR  0500  CSEG
. . . . . .L NEAR  049D  CSEG
. . . . . .L NEAR  04C4  CSEG
. . . . . .L NEAR  04AD  CSEG
. . . . . .L NEAR  04C4  CSEG
. . . . . .L NEAR  04C4  CSEG
. . . . . .L NEAR  0119  CSEG
. . . . . .L NEAR  04DE  CSEG
. . . . . .L NEAR  04E1  CSEG
. . . . . .L NEAR  04F1  CSEG
. . . . . .L NEAR  0513  CSEG
. . . . . .L NEAR  0520  CSEG
. . . . . .L NEAR  053A  CSEG
. . . . . .L NEAR  052A  CSEG
. . . . . .L NEAR  0557  CSEG
. . . . . .L NEAR  0557  CSEG
. . . . . .L NEAR  0146  CSEG
. . . . . .L NEAR  0562  CSEG
. . . . . .L NEAR  0598  CSEG
. . . . . .L NEAR  034B  CSEG
. . . . . .L NEAR  0596  CSEG
. . . . . .L NEAR  0573  CSEG
. . . . . .L NEAR  0596  CSEG
. . . . . .L NEAR  057D  CSEG
. . . . . .L NEAR  0596  CSEG
. . . . . .L NEAR  0587  CSEG
. . . . . .L NEAR  059B  CSEG
. . . . . .L NEAR  05FA  CSEG
. . . . . .L NEAR  05AE  CSEG
. . . . . .L NEAR  05C7  CSEG
. . . . . .L NEAR  05EC  CSEG
. . . . . .L NEAR  05E8  CSEG
. . . . . .L NEAR  05FE  CSEG
. . . . . .L NEAR  060C  CSEG
. . . . . .L NEAR  060E  CSEG
. . . . . .L NEAR  0616  CSEG
. . . . . .L NEAR  015D  CSEG
. . . . . .L NEAR  061D  CSEG
. . . . . .L NEAR  0635  CSEG
. . . . . .L NEAR  0631  CSEG
. . . . . .L NEAR  068B  CSEG
. . . . . .L NEAR  0676  CSEG
. . . . . .L NEAR  0673  CSEG
. . . . . .L NEAR  065D  CSEG
. . . . . .L NEAR  0670  CSEG
```

. L	NEAR	068B	CSEG
. L	NEAR	0685	CSEG
. L	NEAR	06CE	CSEG
. L	NEAR	06D7	CSEG
. L	NEAR	06F9	CSEG
. L	NEAR	06F7	CSEG
. L	NEAR	016F	CSEG
. L	NEAR	0766	CSEG
. L	NEAR	072C	CSEG
. L	NEAR	0710	CSEG
. L	NEAR	072C	CSEG
. L	NEAR	0728	CSEG
. L	NEAR	0740	CSEG
. L	NEAR	0758	CSEG
. L	NEAR	0756	CSEG
. L	NEAR	076E	CSEG
. L	NEAR	0796	CSEG
. L	NEAR	07A2	CSEG
. L	NEAR	07AF	CSEG
. L	NEAR	07D3	CSEG
. L	NEAR	07C0	CSEG
. L	NEAR	00B5	CSEG
. L	NEAR	07D8	CSEG
. L	NEAR	018A	CSEG
. L	NEAR	07F9	CSEG
. L	NEAR	082A	CSEG
. L	NEAR	0827	CSEG
. L	NEAR	0832	CSEG
. L	NEAR	084A	CSEG
. L	NEAR	0868	CSEG
. L	NEAR	0868	CSEG
. L	NEAR	088C	CSEG
. L	NEAR	08AD	CSEG
. L	NEAR	08C3	CSEG
. L	NEAR	08C3	CSEG
. L	NEAR	08F3	CSEG
. L	NEAR	018A	CSEG
. L	NEAR	090D	CSEG
. L	NEAR	0937	CSEG
. L	NEAR	097E	CSEG
. L	NEAR	0955	CSEG
. L	NEAR	0979	CSEG
. L	NEAR	0992	CSEG
. L	NEAR	0992	CSEG
. L	NEAR	09AC	CSEG
. L	NEAR	09C2	CSEG
. L	NEAR	09CB	CSEG
. L	NEAR	018A	CSEG
. L	NEAR	0A03	CSEG
. L	NEAR	0A12	CSEG
. L	NEAR	0A21	CSEG
. L	NEAR	0A74	CSEG
. L	NEAR	0A52	CSEG
. L	NEAR	0A4F	CSEG

. L	NEAR	0A69	CSEG
. L	NEAR	0A83	CSEG
. L	NEAR	0ABB	CSEG
. L	NEAR	0A9F	CSEG
. L	NEAR	0AB5	CSEG
. L	NEAR	0AB3	CSEG
. L	NEAR	01B5	CSEG
. L	NEAR	0AD7	CSEG
. L	NEAR	0ADF	CSEG
. L	NEAR	0B26	CSEG
. L	NEAR	0B0E	CSEG
. L	NEAR	0B0B	CSEG
. L	NEAR	0B50	CSEG
. L	NEAR	0B6A	CSEG
. L	NEAR	0B68	CSEG
. L	NEAR	0B66	CSEG
. L	NEAR	0BA8	CSEG
. L	NEAR	0B93	CSEG
. L	NEAR	0B7D	CSEG
. L	NEAR	0B8D	CSEG
. L	NEAR	0B97	CSEG
. L	NEAR	0BA6	CSEG
. L	NEAR	0BFB	CSEG
. L	NEAR	0BEA	CSEG
. L	NEAR	01B3	CSEG
. L	NEAR	0C35	CSEG
. L	NEAR	0C52	CSEG
. L	NEAR	0C51	CSEG
. L	NEAR	0C6C	CSEG
. L	NEAR	0CBD	CSEG
. L	NEAR	0CAC	CSEG
. L	NEAR	0CCD	CSEG
. L	NEAR	0D04	CSEG
. L	NEAR	0D14	CSEG
. L	NEAR	0D1B	CSEG
. L	NEAR	0D66	CSEG
. L	NEAR	0D64	CSEG
. L	NEAR	0D5E	CSEG
. L	NEAR	01A8	CSEG
. L	NEAR	0D57	CSEG
. L	NEAR	01B2	CSEG
. L	NEAR	0D72	CSEG
. L	NEAR	0D8F	CSEG
. L	NEAR	0DC6	CSEG
. L	NEAR	0DF9	CSEG
. L	NEAR	0E10	CSEG
. L	NEAR	0E1B	CSEG
. L	NEAR	0E68	CSEG
. L	NEAR	0E87	CSEG
. L	NEAR	0EF0	CSEG
. L	NEAR	0ECD	CSEG
. L	NEAR	0F04	CSEG
. L	NEAR	0F19	CSEG
. L	NEAR	0F2C	CSEG

```
. . . . . .L NEAR  01CF    CSEG
. . . . . .L NEAR  0F34    CSEG
. . . . . .L NEAR  0F3C    CSEG
. . . . . .L NEAR  0F68    CSEG
. . . . . .L NEAR  0F84    CSEG
. . . . . .L NEAR  0F8F    CSEG
. . . . . .L NEAR  0F9A    CSEG
. . . . . .L NEAR  0FC7    CSEG
. . . . . .L NEAR  1001    CSEG
. . . . . .L NEAR  103B    CSEG
. . . . . .L NEAR  1039    CSEG
. . . . . .L NEAR  01C7    CSEG
. . . . . .L NEAR  10EB    CSEG
. . . . . .L NEAR  01CC    CSEG
. . . . . .L NEAR  10EB    CSEG
. . . . . .L NEAR  10E9    CSEG
. . . . . .L NEAR  10DC    CSEG
. . . . . .L NEAR  1105    CSEG
. . . . . .L NEAR  1105    CSEG
. . . . . .L NEAR  1132    CSEG
. . . . . .L NEAR  117C    CSEG
. . . . . .L NEAR  117C    CSEG
. . . . . .L NEAR  116F    CSEG
. . . . . .L NEAR  11A0    CSEG
. . . . . .L NEAR  01DF    CSEG
. . . . . .L NEAR  01F5    CSEG
. . . . . .Number  0000
. . . . . .Number  22F6
. . . . . .Number  22F8
. . . . . .Number  22F6
. . . . . .Number  22C4
. . . . . .Number  2198
. . . . . .Number  21CA
. . . . . .Number  0258
. . . . . .Number  0002
. . . . . .Number  0002
. . . . . .Number  0003
. . . . . .Number  0002
. . . . . .Number  0002
. . . . . .Number  0002
. . . . . .Number  FFFF
. . . . . .N PROC  0E0A    CSEG    Length =003A
MSG. . . . . .L BYTE  0DFA    CSEG
N. . . . . .L BYTE  0C51    VSEG
. . . . . .L BYTE  1003    NIC_SEGMENT     Length =0003
. . . . . .L BYTE  0C9F    VSEG
_TYPE. . . . . .Number  0094
. . . . . .L BYTE  005F    VSEG    Length =008F
_FULL. . . . . .L BYTE  0054    VSEG
_PREV. . . . . .L BYTE  00EE    VSEG    Length =008F
E. . . . . .Number  008F
FPTR . . . . . .L WORD  0905    VSEG
FSIZ . . . . . .L WORD  0903    VSEG
NS . . . . . .L BYTE  0907    VSEG
```

```
_END. . . . . .Number  0082
_HEAD . . . . . .Number  001A
_START. . . . . .Number  0080
_TAIL . . . . . .Number  001C
. . . . . .Number  0061
PKR . . . . . .Number  0003
. . . . . .L BYTE  0055    VSEG
SY. . . . . .L BYTE  0020    VSEG
XT. . . . . .L NEAR  0F63    CSEG
. . . . . .N PROC  00B5    CSEG    Global  Length =0047
. . . . . .L BYTE  005D    VSEG
CAN . . . . . .L BYTE  005E    VSEG
. . . . . .Number  5300
. . . . . .Number  5000
. . . . . .Number  4F00
CTIVE . . . . . .Number  0001
DLE . . . . . .Number  0002
. . . . . .Number  3B00
. . . . . .Number  3D00
. . . . . .Number  4700
. . . . . .Number  5200
. . . . . .Number  0016
XIT . . . . . .L NEAR  1027    CSEG
TN. . . . . .F PROC  0F23    CSEG    Global  Length =015F
. . . . . .Number  4B00
S . . . . . .Number  0062
. . . . . .Number  5100
. . . . . .Number  4900
. . . . . .Number  0000
. . . . . .Number  4D00
. . . . . .Number  0002
. . . . . .Number  0001
. . . . . .Number  0063
SEND. . . . . .Number  0061
START . . . . . .Number  005F
STOP. . . . . .Number  0060
. . . . . .Number  4800
. . . . . .L WORD  0CA0    VSEG
. . . . . .L DWORD 0000    CSEG    Global  Length =000E
ISY . . . . . .L BYTE  001F    VSEG
N. . . . . .F PROC  1106    CSEG    Global  Length =0034
ED . . . . . .L NEAR  005B    CSEG
_SEG. . . . . .Number  0000
. . . . . .L NEAR  0049    CSEG
T. . . . . .L NEAR  005E    CSEG
. . . . . .Number  0000
. . . . . .L DWORD 0C66    VSEG
RCV. . . . . .L BYTE  0C5B    VSEG
. . . . . .Number  000A
. . . . . .L BYTE  0911    VSEG
_TYPE. . . . . .Number  0097
TYPE . . . . . .Number  0098
. . . . . .Alias   OFF_L4GET_PTRS
. . . . . .Number  0010
```

```
T . . . . . .Number  00C8
E . . . . . .Number  0010
T . . . . . .Number  0258
. . . . . .L NEAR  0081    CSEG
. . . . . .L BYTE  0000    NIC_SEGMENT     Length =1000
. . . . . .Number  0000
. . . . . .L NEAR  031E    CSEG
. . . . . .L NEAR  01E3    CSEG
. . . . . .Number  0005
. . . . . .Number  0001
. . . . . .Number  0004
. . . . . .Number  0003
ONN . . . . . .Number  0009
. . . . . .Number  0006
E_RB. . . . . .Number  0001
. . . . . .Number  0007
. . . . . .Number  0008
. . . . . .Number  0004
. . . . . .Number  000C
TRS . . . . . .Number  000D
. . . . . .Number  0000
. . . . . .Number  0003
CV. . . . . .Number  0005
G . . . . . .Number  000B
_RB . . . . . .Number  0002
G . . . . . .Number  000A
TOR . . . . . .L DWORD 0C75    VSEG
. . . . . .L DWORD 0C6C    VSEG
C . . . . . .L DWORD 0C79    VSEG
. . . . . .Number  0012
. . . . . .Number  0000
. . . . . .L BYTE  0026    VSEG
BUSY. . . . . .L BYTE  0021    VSEG
. . . . . .L BYTE  0C59    VSEG
G . . . . . .L WORD  0C6A    VSEG
E . . . . . .L WORD  002E    VSEG
. . . . . .L BYTE  0029    VSEG
. . . . . .L NEAR  0B1B    CSEG
S . . . . . .N PROC  0AC1    CSEG    Length =0075
_MSG. . . . . .N PROC  0599    CSEG    Length =0075
. . . . . .N PROC  00FC    CSEG    Length =028C
RET . . . . . .L NEAR  0387    CSEG
. . . . . .N PROC  055E    CSEG    Length =003B
_Z. . . . . .L NEAR  0598    CSEG
_MSG. . . . . .N PROC  060E    CSEG    Length =00B0
_MSG. . . . . .N PROC  06FD    CSEG    Length =006A
D . . . . . .L NEAR  0697    CSEG
D . . . . . .L NEAR  05EE    CSEG
. . . . . .L WORD  0C73    VSEG
. . . . . .Number  002C
. . . . . .L NEAR  07EB    CSEG
. . . . . .L BYTE  0C56    VSEG
. . . . . .L BYTE  017D    VSEG    Length =03C3
. . . . . .Number  03C3
```

```
N. . . . . .L NEAR  082D    CSEG
 . . . . . .L NEAR  0892    CSEG
PE . . . . . .L BYTE  0C5A    VSEG
 . . . . . .L DWORD 0C5E    VSEG
 . . . . . .N PROC  0508    CSEG    Global  Length =0056
AIN. . . . . .L NEAR  07F0    CSEG
 . . . . . .L BYTE  001D    VSEG
GE . . . . . .N PROC  0767    CSEG    Length =0164
 . . . . . .L NEAR  076E    CSEG
_RB. . . . . .L NEAR  0E5D    CSEG
S. . . . . .N PROC  0E44    CSEG    Global  Length =0025
 . . . . . .Number  0002
 . . . . . .L WORD  1006    NIC_SEGMENT     Length =0003
S. . . . . .L WORD  100C    NIC_SEGMENT
 . . . . . .L BYTE  1002    NIC_SEGMENT
_TYPE. . . . . .Number  0096
 . . . . . .Number  0080
T. . . . . .L BYTE  0915    VSEG
 . . . . . .L WORD  0013    VSEG
 . . . . . .L WORD  0011    VSEG
YPE. . . . . .Number  0095
N. . . . . .Number  0601
N. . . . . .Number  0604
N. . . . . .Number  0603
N. . . . . .Number  0602
 . . . . . .L NEAR  0C48    CSEG
 . . . . . .L BYTE  001C    VSEG
 . . . . . .L NEAR  0A17    CSEG
G. . . . . .N PROC  0995    CSEG    Length =012C
GE . . . . . .N PROC  0BA9    CSEG    Length =00BF
TR . . . . . .L DWORD 0057    VSEG
QST. . . . . .L BYTE  0027    VSEG
TAT. . . . . .L BYTE  005C    VSEG
TN . . . . . .L BYTE  005B    VSEG
 . . . . . .L NEAR  0A69    CSEG
UNT. . . . . .L BYTE  0C53    VSEG
UNT. . . . . .L BYTE  0C52    VSEG
N. . . . . .L NEAR  084B    CSEG
P. . . . . .L NEAR  0872    CSEG
 . . . . . .N PROC  08CB    CSEG    Global  Length =00CA
S. . . . . .Alias   C_TRUE
ONS. . . . . .N PROC  0397    CSEG    Length =0171
OLD. . . . . .L NEAR  04B6    CSEG
RPMSG. . . . . .L BYTE  0388    CSEG
RPMSGL . . . . . .Number  0008
EXT. . . . . .L NEAR  04FE    CSEG
TNS. . . . . .L NEAR  042D    CSEG
SRMSG. . . . . .L BYTE  0390    CSEG
SRMSGL . . . . . .Number  0007
 . . . . . .L NEAR  0500    CSEG
 . . . . . .L BYTE  0C57    VSEG
 . . . . . .L BYTE  0540    VSEG    Length =03C3
_SIZE. . . . . .L WORD  0908    VSEG
_TXT . . . . . .L WORD  090A    VSEG
```

```
E. . . . . .Number  03C3
 . . . . . .L DWORD 0C62    VSEG
E. . . . . .Number  0003
OUND . . . . . .Number  0001
 . . . . . .Number  0000
 . . . . . .L WORD  0F00    VSEG
 . . . . . .L WORD  1158    VSEG
 . . . . . .L WORD  13B0    VSEG
 . . . . . .L WORD  1608    VSEG
 . . . . . .N PROC  0C84    CSEG    Length =003C
 . . . . . .L BYTE  0914    VSEG
 . . . . . .L NEAR  0C81    CSEG
 . . . . . .L BYTE  0916    VSEG    Length =0258
IZE. . . . . .L WORD  0912    VSEG
 . . . . . .N PROC  0C68    CSEG    Length =001C
UND. . . . . .L NEAR  0C83    CSEG
IZE. . . . . .Number  0005
 . . . . . .N PROC  0CC0    CSEG    Length =0037
E. . . . . .Number  00AA
 . . . . . .L BYTE  0028    VSEG
M_ROW. . . . . .L BYTE  0044    VSEG
COL. . . . . .L BYTE  0041    VSEG
RIGHT. . . . . .L WORD  0043    VSEG
_COL . . . . . .L BYTE  0043    VSEG
OW . . . . . .L BYTE  0042    VSEG
LEFT . . . . . .L WORD  0041    VSEG
 . . . . . .N PROC  11A3    CSEG    Global  Length =003A
 . . . . . .Number  0D04
 . . . . . .L WORD  0007    VSEG
 . . . . . .L WORD  0009    VSEG
 . . . . . .L WORD  000B    VSEG
 . . . . . .L WORD  000D    VSEG
VE . . . . . .L BYTE  001B    VSEG
 . . . . . .L WORD  0007    VSEG
 . . . . . .Number  0002
 . . . . . .L BYTE  002A    VSEG
T. . . . . .Number  0006
 . . . . . .Number  0002
 . . . . . .Number  0000
 . . . . . .Number  0005
N. . . . . .Number  0001
L. . . . . .Number  0008
ROR. . . . . .Number  0004
FL . . . . . .Number  0007
RROR . . . . . .Number  0003
 . . . . . .L WORD  0C7D    VSEG
 . . . . . .Number  0006
 . . . . . .Number  0042
 . . . . . .Number  0043
_MSB . . . . . .Number  00A6
 . . . . . .Number  001A
BUSY . . . . . .L BYTE  001E    VSEG
RTN. . . . . .F PROC  113A    CSEG    Global  Length =0069
 . . . . . .Number  0000
```

```
. . . . . .Number  0008
. . . . . .L BYTE  002C    VSEG
CKS . . . . . .Number  0002
. . . . . .N PROC  0DBE    CSEG    Global  Length =000B
. . . . . .Number  0001
. . . . . .L BYTE  0C81    VSEG
. . . . . .L BYTE  0C85    VSEG
. . . . . .L BYTE  0C89    VSEG
. . . . . .L BYTE  0C8B    VSEG
. . . . . .L BYTE  0C8F    VSEG
. . . . . .L BYTE  0C92    VSEG
. . . . . .L WORD  0C7F    VSEG
_ROW. . . . . .L BYTE  004C    VSEG
OL. . . . . .L BYTE  0049    VSEG
IGHT. . . . . .L WORD  004B    VSEG
COL . . . . . .L BYTE  004B    VSEG
W . . . . . .L BYTE  004A    VSEG
EFT . . . . . .L WORD  0049    VSEG
. . . . . .Number  00CC
RSOR. . . . . .Number  0003
. . . . . .Number  0010
L . . . . . .Number  0006
RSOR. . . . . .Number  0002
_CH . . . . . .Number  000A
_TTY. . . . . .Number  000E
. . . . . .L NEAR  0000    External
N . . . . . .L BYTE  0022    VSEG
ERS . . . . . .L BYTE  160F    VSEG
E . . . . . .Number  0002
NE. . . . . .Number  0000
URS . . . . . .Number  0004
. . . . . .Number  0007
. . . . . .Number  0001
OP. . . . . .Number  0009
T . . . . . .Number  0003
CT. . . . . .Number  0006
TTR . . . . . .Number  0008
URS . . . . . .Number  0005
. . . . . .L WORD  0C5C    VSEG
. . . . . .N PROC  0D92    CSEG    Length =0008
. . . . . .N PROC  0D9A    CSEG    Length =0012
BLE . . . . . .L NEAR  0DB9    CSEG
. . . . . .N PROC  0DAC    CSEG    Length =0012
. . . . . .Number  0003
. . . . . .Number  0002
. . . . . .Number  0000
. . . . . .Number  0005
. . . . . .Number  0001
. . . . . .Number  0004
. . . . . .L NEAR  014B    CSEG
```

-ee

-e
-s