

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 16:52:23

Print request number 808

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

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

;-----;
;       TALK initialization.
;
;           Everything above here stays resident.
;           Everything below here is transient.
;
;           See the TALKC.ASM and TALKH.ASM files for lots more information
;           about TALK and how it's organized.
;
;-----;

        .sall          ;suppress macro expansions

        subttl ...Transient initialization code...

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

C           include talkv.asm           ;symbols and variables
C
C ;-----;
C ;   talkv.asm    TALK symbols and variables
C ;
C ;-----;
C
C version      macro
C db           '0.48'                 ;version number
C endm

= 0D04
C talk_wks      equ      0d04h ;+channel!;well-known socket base. Change it
C                                     ;whenever packet format changes!
C
C           .lfccnd
C c_true        equ      0ffffh
C c_false       equ      0000h
C
C show_rumors   equ      c_true      ;show rumored stns in str_list?
C assert        equ      c_true      ;generate internal error checking?
C
C           extrn  window:near
C
C cgrp         group  hseg,vseg,cseg
C
= FFFF
C hseg          segment common 'hsegcl' ;dummy (this assembler'll kill me yet)
C ends
C
= 0000
C vseg          segment common 'vsegcl'
```

...Transient initialization code...

```
C
C ;
C ; LEVEL 4 SYMBOLS
C ;
C
= 0000    C on_nic      equ     0
= 0000    C 14_in_our_seg equ     0
C
C ;; include e:14asm.itf           ;level 4 interface
C .list
C
C ;
C ; SYMBOLS
C ;
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_nwccmds, 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
= 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
= 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
= 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
= 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
= 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
= 4800 C key_left equ 4b00h
= 4d00 C key_right equ 4d00h
C
= 3800 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
= 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
= 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
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 ;           LOCAL STORAGE
C ;
C 0000 03 14 15 54 41 4C          C db      3h,14h,15h,'TALK'       ;debug marker (pi)
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
C 000F 0000        C curtask     dw      0           ;current TCB offset
C
C 0011 ????       C savss       dw      ?           ;keyboard int stack
C 0013 ????       C savsp       dw      ?
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_recvfd   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
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 ;      Windows
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
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
C
0045          C  dwin_upperleft  label word
0045 05          C  dwin_left_col db      5
;upper left corner of display window
0046 06          C  dwin_top_row db      6
0047          C  dwin_lowerright label word
0047 4E          C  dwin_right_col db     78
;lower right corner of display window
0048 14          C  dwin_bottom_row db     20
C
C
0049          C  twin_upperleft  label word
0049 05          C  twin_left_col db      5
;upper left corner of typing window
004A 14          C  twin_top_row db     20
004B          C  twin_lowerright label word
004B 4E          C  twin_right_col db    78
;lower right corner of typing window
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
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
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
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
00EE 8F [        C  kbd_buffer_prev db      kbd_bufsize dup(?) ;previous keyboard buffer
??              ]
C
017D 03C3 [      C  rcv_buffer db      rcv_bufsize dup(?) ;receive message buffer
??              ]
C
0540 03C3 [      C  snd_buffer db      snd_bufsize dup(?) ;send message buffer
??              ]
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/r/t previous kbd line
C
0908 ????
090A ????
C
C ;       Station and group lists
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.
C ; Search for "stn_" or "grp_" to find those occurrences.
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
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
0000 ??          C stn_rec      struc
0001 ??          C stn_addr    db     ?           ;station addr, or zero if end
0002 ????
0004 ??          C stn_flags    db     ?           ;station flags -- see below
0005 ?? ??        C stn_groupmask dw    ?           ;group membership mask
0007
= 0005          C stn_name1   db     ? ,? ;...      ;length of following name
          C stn_name    db     ? ,? ;...      ;station name
          C stn_rec     ends
          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
0B70 01
C grp_list_size dw 5 ;current size of group list, +0
C grp_count db 1 ;count of groups
C
0B71 03 41 4C 4C 00
0B76 C8 [ ?? ]
C grp_list db 3,"ALL",0 ;group list, with records:
C db max_grp_list dup (?)
C
C
0000 ?? grp_rec struct
0001 ?? ?? grp_name1 db ? ;length of name
0003 grp_name db ?,?;... ;group name
= 0001 grp_rec ends
C grp_rec_fsize equ 1 ;length of fixed part of grp_rec
C
0C3E 11 [ 00
C grp_number_map db max_groups+1 dup (0) ;group number map[1..16]
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 dest_grp db 1 ;relative group number
0C50 01 dest_stn db 1 ;relative station number
0C51 00 is_dest_stn db 0 ;1 if destination is stn
C
0C52 ?? send_ok_count db ? ;count of successful sends
0C53 ?? send_ng_count db ? ;count of unsuccessful sends
C
C ; Network stuff
C
0C54 4E 57
0C56 ?? rcv_arclnet db 'NW' ;debug marker
0C57 ?? snd_arclnet db ? ;arclnet addr of stn rcvd from
0C58 00 brdcst_arclnet db 0 ;arclnet addr of stn sent to
0C59 ?? our_arclnet db ? ;arclnet addr of stn which broadcast
0C5A ?? rcv_msg_type db ? ;our arclnet address
0C5B ?? last_open_rcv db ? ;incoming message type
C
0C5C 0D04 wks dw talk_wks ;last time we did open_recv
C ; (low bits of clock only)
;our well-known socket
; (modified by channel number)
0C5E 00 00 00 00 rcv_rb dd 0 ;ptrs to transport level
0C62 00 00 00 00 snd_rb dd 0 ;request blocks (rbs)
0C66 ????????
0C6A 0000 our_14_debug dw 0 ;ptr to L4 publics
0C6C ???????? old_14_exit dd ? ;the 14 debug value
;existing 14 exit rtn
```

```
C
C
C ; DOS stuff
C
C
0C70 44 4F 53 C db 'DOS' ;debug marker
0C73 ??? C 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
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 beep 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 tweedle6 db 15,15,15,11,11,11,0
00
C
0C9F 0A C kb10 db 10
0CA0 0444 C kw1092 dw 1092 ;18.2 * 60 for time conversion
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
0F00 0258 [ C stack0 label word
0F00 0258 [ C db 600 dup(71h) ;1: keyboard process
    71
]
C
1158 0258 [ C stack1 label word
1158 0258 [ C db 600 dup(72h) ;2: receive process
    72
]
C
13B0 0258 [ C stack2 label word
13B0 0258 [ C db 600 dup(73h) ;3: send process
    73
]
```

```
] 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.
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
C 53
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 ;           -----      -----      -----
C ;           db          1          * (2-0+1)  * (78-5+1) * 2 dup (?) ;the group window
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
3953      C vseg          ends
C
C
0000      C cseg          segment public 'code'
C                         assume cs:cgrp,es:nothing,ds:nothing
C
C                         public talkinit
```

level_four_interface

```
        extrn  parm_len:byte, parm_str:byte           ;in talkh
        extrn  send_task:near,rcv_task:near,kb_task:near   ;in talkc
        extrn  key_int_rtn:far,timer_int_rtn:far          ;in talkc
        extrn  to_hex:near,release_rbs:near,talk_exit:near ;in talkc
        extrn  14_entries:dword,al4locate:near,14_exit_rtn:far ;in talkc

0000          talkinit      proc    near
0000  FC          cld
0001  2E: 8C 1E 0C73 R      mov    psp,ds           ;direction forward
0006  8C C8          mov    ax,cs           ;save the addr of the dos PSP
0008  8E C0          mov    es,ax           ;es:=cs
000A  8E D8          mov    ds,ax           ;ds:=cs
000B          assume  ds:cgrp,es:cgrp

;       Check if TALK is already installed and exit if so

000C  B4 63          mov    ah,key_talk
000E  CD 16          int    key_int           ;special extended function
0010  80 FC 9C          cmp    ah,not key_talk
0011          $if    e
0015  E9 0525 R      jmp    error_install
0016          $endif

%out     ...command line parsing

;
;       Parse the command line parameters.
;
;       ds:si is the current pointer, and cx the current length.
;

0018  8A 0E 0000 E      mov    cl,parm_len      ;parm length
001C  B5 00          mov    ch,0
001E  8D 36 0000 E      lea    si,parm_str      ;offset of parms
0022  E8 076E R      call   str_skip_blanks   ;skip leading blanks
0025  E8 0757 R      call   str_upper         ;convert to upper case

0028          scan_keywords:
0028  8D 3E 045A R      lea    di,qdebug        ;check for "DEBUG" -----
002C  E8 0779 R      call   str_scan
002D          $ifnot c
0031  C7 06 0C6A R 0010      mov    our_14_debug,10h;allow debugging messages
0037  EB EF          jmp    scan_keywords
0038          $endif

0039  8D 3E 0460 R      lea    di,qtest         ;check for "TEST" -----
003D  E8 0779 R      call   str_scan
003E          $ifnot c
0042  C6 06 002A R 01      mov    test_mode,1
0047  EB DF          jmp    scan_keywords
0048          $endif
```

```
0049 8D 3E 0450 R          lea    di,qtimestamp      ;check for "TIMESTAMP" -----
004D E8 0779 R          call   str_scan
$ifnot
        mov   timestamp,1
        jmp   scan_keywords
$endif

0052 C6 06 002C R 01
0057 EB CF

0059 8D 3E 0465 R          lea    di,qkey           ;check for "KEY" -----
005D E8 0779 R          call   str_scan
$ifnot
        c
        call  str_to_num       ;get character number
$endif

0062 E8 07E0 R          $if c
                           lea    di,qf             ;check for 'Fn'
                           call  str_scan
$if
        c
        jmp   error_key
$endif

0067 8D 3E 0469 R          lea    di,qf             ;no number,
006B E8 0779 R          call   str_scan
$if
        c
        jmp   error_key
$endif

0070 E9 051B R          j_error_key:   jmp   error_key
$endif

0073 E8 07E0 R          call  str_to_num       ;get n in {1..10}
0076 72 F8
0078 48
0079 78 F5
007B 3D 0009
007E 77 F0
0080 04 3B
0082 A2 005E R          add   al,59           ;create 59..68 extended code
0085 C6 06 005D R 00
                           mov   keychar_scan,al
                           mov   keychar,0

$else
008D A2 005D R          mov   keychar,al       ;got a number....
0090 0A C0
                           or    al,al           ;store trigger char
$if
        z
        call  str_to_num       ;extended code?
$endif
                           jc    j_error_key
                           mov   keychar_scan,al
                           ;yes...
                           ;get second number
                           ;and save it
                           ;extended code

0094 E8 07E0 R          $endif
                           jmp   scan_keywords
                           ;"KEY" parameter

009C EB 8A
$endif

009E 8D 3E 046B R          lea    di,qpopup          ;check for "POPUP" -----
00A2 E8 0779 R          call   str_scan
$ifnot
        c
        mov   auto_popup,1
        jmp   scan_keywords
$endif

00A7 C6 06 002B R 01
00AC E9 0028 R

00AF 8D 3E 0471 R          lea    di,qpopup          ;check for "POPDOWN" -----
00B3 E8 0779 R          call   str_scan
```

```

00B8 C6 06 002D R 01
00BD E8 07E0 R
00C2 3D 0258
00C7 E9 052F R
00CA F7 26 0492 R
00CE A3 002E R
00D1 E9 0028 R

$ifnot c
    mov auto_popupdown,1
    call str_to_num
    $ifnot c
        cmp ax,10*60
        $if a
            jmp error_popupdown
        $endif
        mul k18
        mov popdown_time,ax
    $endif
    jmp scan_keywords
$endif

00D4 8D 3E 0479 R
00D8 E8 0779 R
00DD E8 07E0 R
00E2 E9 0534 R
00E5 3D 000A
00EA E9 0534 R
00ED A2 0030 R
00F0 05 0D04
00F3 A3 0C5C R
00F6 E9 0028 R

lea di,qchannel
call str_scan
$ifnot c
call str_to_num
$if c
jmp error_channel
$endif
cmp ax,10
$if ae
jmp error_channel
$endif
mov channel,al
add ax,talk_wks
mov wks,ax
jmp scan_keywords
$endif

;check for "CHANNEL" -----
;must supply a channel number
;range is 0..9
;add channel to base wks
;save well-known socket

00F9 8D 3E 0487 R
00FD E8 0779 R
0102 E8 07E0 R
0107 E9 053E R
j_error_userwind: jmp error_userwind
$endif
cmp ax,10
ja j_error_userwind
or ax,ax
jz j_error_userwind
add al,swin_top_row
inc al
mov swin_bottom_row,al
mov dwin_top_row,al
jmp scan_keywords
$endif

;check for "USERWINDOW" -----
;must supply number of lines
;range is 1..10
;compute new bottom row of user window
;(including bottom border)
;store as bottom row of user window
;and as top row of display window

0122 8D 3E 0481 R
0126 E8 0779 R

lea di,qcolor
call str_scan
;check for "COLOR" -----

```

```
012B E8 07E0 R           $ifnot c          ;must supply a color modifier
                           call str_to_num
                           $if c
                           jmp error_color
                           $endif
                           cmp ax,255          ;range is 0..255
                           $if ae
                           jmp error_color
                           $endif
                           mov color,al          ;save the color modifier
                           xor char_normal,al
                           xor char_bold,al
                           xor char_line,al
                           xor char_boldline,al
                           jmp scan_keywords
                           $endif

; Add other parameter parsing here... -----  

0151 E3 08                jcxz parms_done      ;check for nothing else on
0153 80 3C 0D              cmp byte ptr[si],cr    ; the command line
                           $ifnot e
                           jmp error_parms
                           $endif
0158 E9 0520 R
015B                         parms_done:  

                           %out ...var initialization

; Initialize the task stacks
015B B9 0000 E             mov cx,offset cgrp:kb_task      ;keyboard task
015E BB 0002
0161 E8 0702 R             mov bx,task1-tcbs
                           call init_task
0164 B9 0000 E             mov cx,offset cgrp:recv_task   ;receive task
                           mov bx,task2-tcbs
                           call init_task
0167 BB 0004
016A E8 0702 R             mov cx,offset cgrp:send_task;send task
                           mov bx,task3-tcbs
                           call init_task

; Define the four windows.
0176 8D 3E 0495 R           lea di,borders+1        ;modify border color attributes
017A B9 0024
017D A0 0031 R             mov cx,4*9          ;4 windows, 9 specifiers for each
                           $do
                           xor [di],al          ;modify it
                           add di,2
                           $repeatloop
```

0187 B4 08	mov ah,window_setattr	;set attribute for scrolled lines
0189 A0 0032 R	mov al,char_normal	
018C E8 0000 E	call window	
018F 8D 3E 160F R	lea di,window_buffers	;di = start of window buffers
0193 B8 0604	mov ax,select_gwin	;---group window---
0196 E8 0000 E	call window	
0199 8B 0E 003D R	mov cx,gwin_upperleft	
019D 8B 16 003F R	mov dx,gwin_lowerright	
01A1 B3 01	mov bl,1	;one page
01A3 B8 000F	mov ax,window_define*256+15	;define it, wrap limit 15
01A6 E8 0000 E	call window	
01A9 8D 36 0494 R	lea si,group_border	
01AD B4 07	mov ah,window_init	;initialize the buffer
01AF E8 0000 E	call window	
01B2 8B F8	mov di,ax	;di = buffer for next window
01B4 B8 0603	mov ax,select_swin	;---station window---
01B7 E8 0000 E	call window	
01BA 8B 0E 0041 R	mov cx,swin_upperleft	
01BE 8B 16 0043 R	mov dx,swin_lowerright	
01C2 B3 01	mov bl,1	;one page
01C4 B8 000F	mov ax,window_define*256+15	;define it, wrap limit 15
01C7 E8 0000 E	call window	
01CA 8D 36 04A6 R	lea si,station_border	
01CE B4 07	mov ah,window_init	;initialize the buffer
01D0 E8 0000 E	call window	
01D3 8B F8	mov di,ax	;di = buffer for next window
01D5 B8 0601	mov ax,select_dwin	;---display window---
01D8 E8 0000 E	call window	
01DB 8B 0E 0045 R	mov cx,dwin_upperleft	
01DF 8B 16 0047 R	mov dx,dwin_lowerright	
01E3 B3 03	mov bl,n_pages	;number of pages
01E5 B8 000F	mov ax,window_define*256+15	;define it, wrap limit 15
01E8 E8 0000 E	call window	
01EB 8D 36 04B8 R	lea si,display_border	
01EF B4 07	mov ah,window_init	;initialize the buffer
01F1 E8 0000 E	call window	
01F4 8B F8	mov di,ax	;di = buffer for next window
01F6 B8 0602	mov ax,select_twin	;---typing window---
01F9 E8 0000 E	call window	
01FC 8B 0E 0049 R	mov cx,twin_upperleft	
0200 8B 16 004B R	mov dx,twin_lowerright	
0204 B3 01	mov bl,1	;one page
0206 B8 000F	mov ax,window_define*256+15	;define it, wrap limit 15
0209 E8 0000 E	call window	
020C 8D 36 04CA R	lea si,typing_border	
0210 B4 07	mov ah,window_init	;initialize the buffer
0212 E8 0000 E	call window	

```
0215 B9 0009 90          mov    cx,l_talk_name           ;put name in upper left corner
0219 8D 36 04FB R        lea    si,talk_name           ;of group window
021D 8D 3E 1613 R        lea    di,group_window+4
$do
0221 A4                 movsb
0222 47                 inc    di                  ;(skip attribute bytes)
$repeatloop

0225 80 3E 0030 R 00      cmp    channel,0            ;if channel # is not default
$ifnot e
022C B9 0008 90          mov    cx,l_channel_name     ;put channel number
0230 8D 36 0504 R        lea    si,channel_name
0234 8A 1E 003F R        mov    bl,gwin_right_col   ;on right of top border
0238 2A 1E 003D R        sub    bl,gwin_left_col    ; of group window
023C 32 FF               xor    bh,bh
023E 81 EB 000A          sub    bx,l_channel_name+2
0242 D1 E3               sal    bx,1 ;(for attributes)
0244 8D BF 160F R        lea    di,group_window[bx]
$do
0248 A4                 movsb
0249 47                 inc    di                  ;(skip attribute)
$repeatloop
024C A0 0030 R          mov    al,channel
024F 04 30               add    al,'0'
0251 AA                 stosb
$endif
0252 B8 0602
0255 E8 0000 E           mov    ax,select_twin           ;default window is typing
call   window
assume ds:nothing,es:nothing

; Initialize the network

025B 3C 00               14_call 14locate           ;find transport-level routines
$ifnot e
025F E9 050C R           jmp    error_nwinit
$endif
0262 2E: 89 3E 0C66 R     mov    word ptr 14_publics,di ;save ptr to L4 publics
0267 2E: 8C 1E 0C68 R     mov    word ptr 14_publics+2,ds
026C 2E: A1 0C6A R        mov    ax,our_14_debug      ;set 14 debugging flag
0270 87 45 06              xchg   ax,ds:[di].14_debug
0273 2E: A3 0C6A R        mov    our_14_debug,ax    ;save old value
0277 8A 5D 05              mov    bl,ds:[di].our_arc
027A 2E: 88 1E 0C59 R     mov    our_arclnet,bl   ;save our arclnet address
027F 2E: 88 1E 0916 R     mov    stn_list.stn_addr,bl ;also as first station
0284 2E: C6 06 0917 R 00  mov    stn_list.stn_flags,0  ;and zero its flags
028A 2E: C7 06 0918 R 8000 mov    stn_list.stn_groupmask,8000h ;only group is "ALL"

0291 2E: A1 0C5C R        mov    ax,wks             ;setup listen on our socket
14_call ignore            ; (clear any old listens first)
```

```
029A 2E: A1 0C5C R          mov     ax,wks
029E B3 01                   mov     bl,1
                                14_call listen
                                cmp     al,sock_ok
$ifnot e
                                jmp     error_socket
$endiff

02AC E8 06DA R          call    get_rb      ;get rcv rb pointer in ds:si
02AF 2E: 89 36 0C5E R      mov     word ptr rcv_rb,si ;save it
02B4 2E: 8C 1E 0C60 R      mov     word ptr rcv_rb+2,ds

02B9 E8 06DA R          call    get_rb      ;get send rb pointer in ds:si
02BC 2E: 89 36 0C62 R      mov     word ptr snd_rb,si ;save it
02C1 2E: 8C 1E 0C64 R      mov     word ptr snd_rb+2,ds

%out   ...parse environment strings

;       Setup our station record, using the name from the "user=xxxx"
;       environment string, if present.

02C6 2E: 8E 1E 0C73 R      mov     ds,psp      ;get the psp segment
02CB 8D 36 0443 R          lea     si,cs:quser
02CF E8 071B R          call    get_env_string ;look for 'USER='
                                .
$ifnot z                  ;if found,
                                mov     si,0      ;move following name
$do
                                mov     al,es:[di]
                                or     al,al
$exitif z
                                call    chk_bad_char ;checking for illegal chars
$if     e
                                jmp     error_name
$endiff
                                mov     stn_list[si].stn_name,al
                                inc     di
                                inc     si
$repeat
                                mov     stn_list[si].stn_name,0 ;end name with zero
                                mov     ax,si      ;length of name
                                cmp     al,max_namesize
$if     g
                                jmp     error_name
$endiff
                                mov     stn_list.stn_name1,al ;save length of name
                                add     si,stn_rec_fsize+1 ;total length of stationlist
                                mov     stn_list_size,si; (including ending zero)

$else
                                mov     stn_list.stn_name,'$' ;else use "$nn" as the name
                                mov     al,our_arcnet ;where "nn" is our station addr
                                mov     cl,4
```

```
0319 D2 C8          ror    al,cl
031B E8 0000 E       call   to_hex
031E 2E: A2 091C R   mov    stn_list.stn_name+1,al
0322 2E: A0 0C59 R   mov    al,our_arclnet
0326 E8 0000 E       call   to_hex
0329 2E: A2 091D R   mov    stn_list.stn_name+2,al
032D 2E: C6 06 091E R 00  mov    stn_list.stn_name+3,0
0333 2E: C6 06 091A R 03  mov    stn_list.stn_name1,3
0339 2E: C7 06 0912 R 0009  mov    stn_list_size,stn_rec_fsize+3+1 ;total stn_list size
$endif

;
; Record any group membership from the "group=xxx,yyy,..."
; environment string, if present.
;

0340 2E: 8E 1E 0C73 R      mov    ds,psp           :get the psp segment
0345 8D 36 0449 R      lea    si,cs:qgroup        :look for 'GROUP='
0349 E8 071B R      call   get_env_string

$ifnot z               ;found; ptr in es:di
034E 8C C0             mov    ax,es
0350 8E D8             mov    ds,ax
0352 8B F7             mov    si,di
0354 8C C8             mov    ax,cs
0356 8E C0             mov    es,ax           ;prepare es for movsb

0358 E8 0757 R         call   str_upper        ;convert to upper case
$do
035B E8 079B R         call   str_name         ;get a name
$exitif c
0360 83 FB 10
0365 E9 052A R
0368 8D 3E 0B70 R      lea    di,grp_list-1     ;point di to zero at the
036C 2E: 03 3E 0B6E R   add    di,grp_list_size; end of the group list
0371 2E: 01 1E 0B6E R   add    grp_list_size,bx;update size of group list
0376 2E: 83 06 0B6E R 01  add    grp_list_size,grp_rec_fsize

037C 8A C3             mov    al,bl           ;size of this name
037E AA               stos   es:grp_list.grp_name1
037F 2B CB             sub    cx,bx           ;how much is left after the name
0381 51               push   cx
0382 8B CB             mov    cx,bx           ;save remaining count
0384 F3/ A4             rep    movsb           ;move the name
0386 B0 00             mov    al,0
0388 AA               stosb
0389 2E: FE 06 0B70 R   inc    grp_count        ;update count of groups
038E 2E: 8A 0E 0B70 R   mov    cl,grp_count        ;update our groupmask
0393 33 DB             xor    bx,bx
```

```
0395 F9          stc
0396 D3 DB        rcr bx,cl
0398 2E: 09 1E 0918 R   or stn_list.stn_groupmask,bx
039D 59          pop cx      ;restore remaining count
                      ;repeat           ;continue for all names
$endif

:               %out ...interrupt installers

;       Print the installation message.
;
; We do this before installing the interrupt vectors so that if the program
; is aborted with ctrl-C, we don't leave the vectors installed without
; staying resident.

03A0 8D 16 04DC R   lea    dx,cs:install_ok_msg
03A4 8C C8          mov    ax,cs
03A6 8E D8          mov    ds,ax      ;ds:dx points to the string
03A8 B4 09          mov    ah,dosint_prints
03AA CD 21          int    dos_int

03AC 2E: C6 06 001B R 01   mov    tasks_active,1      ;set busy flag to keep things
                           ;quiet until all interrupt routines
                           ;are installed

;
;       install the int 16 keyboard interceptor
;

03B2 BE 0000          mov    si,0      ;point es:si at keyboard vector
03B5 8E C6          mov    es,si
03B7 BE 0058          mov    si,key_int*4
03BA B8 0000 E         mov    ax,offset cgrp:key_int_rtn
03BD 8C CB          mov    bx,cs      ;bx:ax is pointer to intercept
03BF 26: 87 04          xchg   ax,es:[si]      ;swap old pointer for new
03C2 26: 87 5C 02          xchg   bx,es:[si+2]
03C6 2E: A3 0C75 R         mov    word ptr old_key_vector,ax      ;save old vector
03CA 2E: 89 1E 0C77 R         mov    word ptr old_key_vector+2,bx

;
;       install the L4 exit routine
;

03CF 2E: C5 36 0C66 R   lds    si,14_publics      ;install 14 exit routine
03D4 B8 0000 E         mov    ax,offset cgrp:14_exit_rtn
03D7 8C CB          mov    bx,cs      ;bx:ax is pointer to exit rtn
03D9 87 44 2E          xchg   ax,ds:[si].user_exit_vector
03DC 87 5C 30          xchg   bx,ds:[si].user_exit_vector+2
03DF 2E: A3 0C6C R         mov    word ptr old_14_exit,ax      ;save old vector
```

```
03E3 2E: 89 1E 0C6E R          mov     word ptr old_14_exit+2,bx

;       install the timer tick interrupt routine
;

03E8 FA                      cli
03E9 BE 0000                  mov     si,0           ;point es:si at interrupt vector
03EC 8E C6                    mov     es,si
03EE BE 0020                  mov     si,timer_tick*4
03F1 B8 0000 E                mov     ax,offset cgrp:timer_int_rtn
03F4 8C CB                    mov     bx,cs          ;bx:ax is pointer to intercept
03F6 26: 87 04                xchg   ax,es:[si]    ;swap old pointer for new
03F9 26: 87 5C 02              xchg   bx,es:[si+2]
03FD 2E: A3 0C79 R            mov     word ptr old_time_vec,ax ;save old vector
0401 2E: 89 1E 0C7B R          mov     word ptr old_time_vec+2,bx
0406 FB                      sti

;       Release the tasks

0407 2E: C6 06 0025 R 01      mov     first_dispatch,1;flag to force a dispatch
040D 2E: C6 06 001B R 00      mov     tasks_active,0        ;clear busy flag

;       If test mode, loop in the foreground reading from the keyboard

0413 2E: F6 06 002A R 01      test   test_mode,1
$ifnot z
$do
041B B4 00                    mov     ah,key_read      ;read a key
041D CD 16                    int    key_int
041F 3C 21                    cmp    al,"!"
$if e
0423 E8 0000 E                call   talk_exit      ; exit
0426 B8 4C00                  mov    ax,dosint_exit*256
0429 CD 21                    int    dos_int         ; and return to dos
$endiff
042B B2 21                    mov    dl,"!"
042D B4 02                    mov    ah,dosint_printc
042F CD 21                    int    dos_int         ;otherwise print bang
$repeat
$endiff                         ;and repeat

;       If not test mode, exit but stay resident

0433 BA 0000 R                mov     dx,offset cgrp:talkinit    ;end of resident part
0436 83 C2 10                add    dx,16          ;round up
0439 B1 04                    mov    cl,4           ;in paragraphs
043B D3 EA                    shr    dx,cl
043D B4 31                    mov    ah,dosint_exitstay
043F B0 00                    mov    al,0           ;exit code
```

0441 CD 21 int dos_int

; Initialization data

0443 05 55 53 45 52 3D quser db 5,"USER="

0449 06 47 52 4F 55 50 qgroup db 6,"GROUP="

3D

0450 09 54 49 4D 45 53 qtimestamp db 9,"TIMESTAMP"

54 41 4D 50

045A 05 44 45 42 55 47 qdebug db 5,"DEBUG"

0460 04 54 45 53 54 QTest db 4,"TEST"

0465 03 4B 45 59 qkey db 3,"KEY"

0469 01 46 qf db 1,"F"

046B 05 50 4F 50 55 50 qpopup db 5,"POPUP"

0471 07 50 4F 50 44 4F qpopdown db 7,"POPDOWN"

57 4E

0479 07 43 48 41 4E 4E qchannel db 7,"CHANNEL"

45 4C

0481 05 43 4F 4C 4F 52 qcolor db 5,"COLOR"

0487 0A 55 53 45 52 57 quserwindow db 10,"USERWINDOW"

49 4E 44 4F 57

0492 0012 k18 dw 18

0494 borders label byte

0494 20 07 C9 07 CD 07 group_border db " ",7,201,7,205,7,187,7,186,7,186,7,199,7,196,7,182,7

BB 07 BA 07 BA 07

C7 07 C4 07 B6 07

04A6 20 07 C7 07 C4 07 station_border db " ",7,199,7,196,7,182,7,186,7,186,7,199,7,205,7,182,7

B6 07 BA 07 BA 07

C7 07 CD 07 B6 07

04B8 20 07 C7 07 C4 07 display_border db " ",7,199,7,196,7,182,7,186,7,186,7,199,7,205,7,182,7

B6 07 BA 07 BA 07

C7 07 CD 07 B6 07

04CA 20 07 C7 07 C4 07 typing_border db " ",7,199,7,196,7,182,7,186,7,186,7,200,7,205,7,188,7

B6 07 BA 07 BA 07

C8 07 CD 07 BC 07

04DC 54 41 4C 4B 20 76 install_ok_msg db 'TALK version '

65 72 73 69 6F 6E

20

04ED 20 69 6E 73 74 61 version

6C 6C 65 64 2E 0D db ' installed.',cr,lf,'\$'

0A 24

04FB 54 41 4C 4B 20 talk_name db 'TALK '

= 0009 l_talk_name equ \$-talk_name

0504 43 68 61 6E 6E 65 channel_name db 'Channel '

6C 20

= 0008 l_channel_name equ \$-channel_name

; Fatal errors

050C BA 0553 R	error_nwinit:	mov dx,offset cgrp:msg_nwinit
050F EB 32		jmp short error_msg
0511 BA 056B R	error_rb:	mov dx,offset cgrp:msg_rb
0514 EB 2D		jmp short error_msg
0516 BA 0585 R	error_socket:	mov dx,offset cgrp:msg_socket
0519 EB 28		short error_msg
051B BA 059B R	error_key:	mov dx,offset cgrp:msg_key
051E EB 23		short error_msg
0520 BA 05B9 R	error_parms:	mov dx,offset cgrp:msg_parms
0523 EB 1E		short error_msg
0525 BA 05E5 R	error_install:	mov dx,offset cgrp:msg_install
0528 EB 19		short error_msg
052A BA 05FF R	error_name:	mov dx,offset cgrp:msg_name
052D EB 14		short error_msg
052F BA 064A R	error_popdown:	mov dx,offset cgrp:msg_popdown
0532 EB 0F		short error_msg
0534 BA 0673 R	error_channel:	mov dx,offset cgrp:msg_channel
0537 EB 0A		short error_msg
0539 BA 0695 R	error_color:	mov dx,offset cgrp:msg_color
053C EB 05		short error_msg
053E BA 06B5 R	error_userwind:	mov dx,offset cgrp:msg_userwind
0541 EB 00		short error_msg
0543 8C C8	error_msg:	mov ax,cs ;ds:dx points to string
0545 8E D8		mov ds,ax
0547 B4 09		mov ah,dosint_prints
0549 CD 21		int dos_int
054B E8 0000 E		call release_rbs ;release any rbs
054E B8 4C10		mov ax,dosint_exit*256+16 ;exit w/ error code 16
0551 CD 21		int dos_int
0553 54 41 4C 4B 3A 20	msg_nwinit	db 'TALK: 14_locate error',cr,lf,'\$'
6C 34 5F 6C 6F 63		
61 74 65 20 65 72		
72 6F 72 0D 0A 24		
056B 54 41 4C 4B 3A 20	msg_rb	db 'TALK: No RBs available.',cr,lf,'\$'
4E 6F 20 52 42 73		
20 61 76 61 69 6C		
61 62 6C 65 2E 0D		
0A 24		
0585 54 41 4C 4B 3A 20	msg_socket	db 'TALK: Bad 14_listen',cr,lf,'\$'
42 61 64 20 6C 34		
5F 6C 69 73 74 65		
6E 0D 0A 24		
059B 54 41 4C 4B 3A 20	msg_key	db 'TALK: Bad key specification',cr,lf,'\$'
42 61 64 20 6B 65		
79 20 73 70 65 63		
69 66 69 63 61 74		

Microsoft MACRO Assembler Version 3.00
TALK - Multiple station conversation program
level_four_interface

Page 1-23
02-15-86

05B9	69 6F 6E 0D 0A 24 54 41 4C 4B 3A 20 55 6E 72 65 63 6F 67 6E 69 7A 65 64 20 63 6F 6D 6D 61 6E 64 20 6C 69 6E 65 20 70 61 72 61 6D 65 74 65 72 0D 0A 24	msg_parms	db	'TALK: Unrecognized command line parameter',cr,lf,'\$'
05E5	54 41 4C 4B 20 61 6C 72 65 61 64 79 20 69 6E 73 74 61 6C 6C 65 64 2E 0D 0A 24	msg_install	db	'TALK already installed.',cr,lf,'\$'
05FF	54 41 4C 4B 3A 20 55 73 65 72 20 6F 72 20 67 72 6F 75 70 20 6E 61 6D 65 20 74 6F 6F 20 62 69 67 20 6F 72 20 0D 0A	msg_name	db	'TALK: User or group name too big or ',cr,lf
0625	20 20 20 20 20 20 63 6F 6E 74 61 69 6E 73 20 69 6E 76 61 6C 69 64 20 63 68 61 72 61 63 74 65 72 73 2E 0D 0A 24		db	' contains invalid characters.',cr,lf,'\$'
064A	54 41 4C 4B 3A 20 50 6F 70 64 6F 77 6E 20 74 69 6D 65 6F 75 74 20 6C 61 72 67 65 72 20 74 68 61 6E 20 36 30 30 2E 0D 0A 24	msg_popup	db	'TALK: Popdown timeout larger than 600.',cr,lf,'\$'
0673	54 41 4C 4B 3A 20 42 61 64 20 63 68 61 6E 65 6C 20 73 70 65 63 69 66 69 63 61 74 69 6F 6E 0D 0A 24	msg_channel	db	'TALK: Bad channel specification',cr,lf,'\$'
0695	54 41 4C 4B 3A 20 42 61 64 20 63 6F 6C 6F 72 20 73 70 65 63 69 66 69 63 61 74 69 6F 6E 0D 0A 24	msg_color	db	'TALK: Bad color specification',cr,lf,'\$'
06B5	54 41 4C 4B 3A 20 42 61 64 20 75 73 65 72 77 69 6E 64 6F 77 20 73 70 65 63 69 66 69 63 61 74 69 6F 6E 0D 0A 24	msg_userwind	db	'TALK: Bad userwindow specification',cr,lf,'\$'

```
06DA          talkinit      endp
              |
              %out    ...utility routines
              |
              ;       get_rb  Get an RB. Fatal exit if none.
              ;       Setup our standard timeouts and retry counts.
              ;       Returns ptr to RB in ds:si.
              ;
06DA          get_rb       proc   near
06DF 8C D8   14_call activate_rb
06E1 0B C0   mov    ax,ds           ;test for valid pointer
              or     ax,ax
              $if   z
              jmp   error_rb        ;no: fatal error
              $endif
06E8 C7 44 48 0006  mov    [si].rb_to_accept_wait,timeout ;accept timeout
06ED C7 44 4C 0006  mov    [si].rb_to_ack_wait,timeout ;ack wait
06F2 C7 44 50 0006  mov    [si].rb_to_pkt_wait,timeout ;packet wait
06F7 C7 44 4E 0002  mov    [si].rb_message_tries,retries ;message tries
06FC C7 44 4A 0002  mov    [si].rb_conn_tries,retries   ;connect tries
0701 C3       ret
0702          get_rb       endp
              |
              |
              ;-----;
              ; INIT_TASK Initialize a task's stack.
              ; BX is the offset of the tcb, CX is the initial IP.
              ;-----;
0702          init_task    proc   near
0702 8B D4   mov    dx,sp           ;save our stack pointer
0704 2E: 8B A7 0007 R  mov    sp,tcbs[bx]    ;get the task's stack pointer
0709 51       push   cx           ;push the return address (initial IP)
070A B8 1234  mov    ax,1234h
070D B9 0009  mov    cx,9          ;push 9 other dummies
              $do
              push   ax
              $repeatloop
0713 2E: 89 A7 0007 R  mov    tcbs[bx],sp    ;save his updated stack pointer
0718 8B E2       mov    sp,dx          ;restore our stack
071A C3       ret
071B          init_task    endp
```

page

```
;-----  
;  
;      get_env_string  
;  
;      Search the DOS environment for a named string.  
;  
; Input: cs:si points to stringname, starting with its length, ending w/ "=".  
;        ds:0 points to the Program Segment Prefix  
; Output: es:di point to the environment string value, terminated w/ 0.  
;        cx is the length of the string.  
;        NZ set if stringname was found, else Z.  
;  
; Destroys: ax.  
;  
;-----  
  
071B          get_env_string    proc    near  
071B 1E          push    ds           ;save ptr to PSP  
071C 8E 06 002C      mov     es,ds:psp_environ ;setup es:di to env strings  
0720 BF 0000      mov     di,0  
0723 8C C8          mov     ax,cs           ;setup ds:si to stringname  
0725 8E D8          mov     ds,ax  
  
          $do          push    si           ;loop for all env. strings  
0727 56          mov     ch,0          ;save start of stringname  
0728 B5 00          mov     cl,ds:[si]  ;get stringname length in cx  
072A 8A 0C          inc     si  
072C 46          repe   cmpsb          ;compare stringname to env.  
072D F3/ A6          pop     si          ;restore start of stringname  
072F 5E          je     get_env_found ;exit if found  
0730 74 12          $do          cmp     byte ptr es:[di],0 ;no match: skip parm value  
0732 26: 80 3D 00      $exitif e  
0738 47          inc     di  
          $repeat          inc     di          ; and ending zero  
073B 47          cmp     byte ptr es:[di],0 ;end of environment?  
073C 26: 80 3D 00      $repeatuntil e          ;no: keep searching  
0742 1F          pop     ds          ;restore ptr to psp  
0743 C3          ret  
  
0744 8B CF          get_env_found:  mov     cx,di           ;found: compute length in cx  
0746 26: 80 3D 00      $do          cmp     byte ptr es:[di],0  
074C 47          $exitif e  
          inc     di  
          $repeat          xchg   cx,di  
074F 87 CF          sub     cx,di  
0751 2B CF          or     al,1           ;set NZ for "found"  
0753 0C 01          pop     ds  
0755 1F          ret  
0756 C3          .           ;and return
```

0757 get_env_string endp

page

```
; str_upper: Convert a string to upper case
;           ds:si  is the start of the string
;           cx     is the length of the string
;           destroys: ax
;

0757  E3 14          str_upper      proc   near
0759  56              jcxz  str_upper_xit
075A  51              push   si
                           push   cx

075B  AC              str_upper_loop: lodsb
075C  3C 61            cmp    al,'a'
075E  72 09            jb    str_upper_nxt
0760  3C 7A            cmp    al,'z'
0762  77 05            ja    str_upper_nxt
0764  2C 20            sub    al, "a" - "A"
0766  88 44 FF          mov    -1[si],al
0769  E2 F0          str_upper_nxt: loop  str_upper_loop

076B  59              pop    cx
076C  5E              pop    si
076D  C3          str_upper_xit: ret
076E          str_upper      endp

;

; str_skip_blanks      Skip leading blanks in a string
;           ds:si  is the start of the string
;           cx     is the length of the string
;

076E  E3 08          str_skip_blanks proc   near
076F          jcxz  str_skip_ret
0770  80 3C 20          str_skip_loop: cmp   byte ptr[si]," "
0773  75 03            jne   str_skip_ret
0775  46              inc   si
0776  E2 F8            loop  str_skip_loop

0778  C3          str_skip_ret: ret
0779          str_skip_blanks endp
```

page

```
; str_scan      Scan a string for a keyword.  
;  
;     ds:si    is the start of the string  
;     cx      is the length of the string  
;     es:di    is the start of the keyword, which begins with a byte length  
;  
;     On return, jnc if the keyword was found, skip following blanks, and  
;     update si,cx for the remainder of the string.  
;  
;     Destroys: ax, di
```

0779	str_scan	proc	near		
0779	53	push	bx		
077A	56	push	si	;save start of string	
077B	51	push	cx	;and its length	
077C	32 FF	xor	bh,bh		
077E	26: 8A 1D	mov	bl,es:[di]	;keyword length from es:di	
0781	47	inc	di		
0782	87 D9	xchq	bx,cx	;keyword length in cx	
0784	2B D9	sub	bx,cx	;remaining string length in bx	
0786	7C 0E	j1	str_scan_false	;string was too short: fail	
0788	F3/ A6	repe	cmpsb	;compare to string	
078A	75 0A	jne	str_scan_false	;no match	
078C	8B CB	mov	cx,bx	;match: return remaining length	
078E	58	pop	ax	;discard saved start/length	
078F	58	pop	ax		
0790	E8 076E R	call	str_skip_blanks	;skip following blanks	
0793	F8	clc		;set success flag	
0794	EB 03	jmp	short str_scan_xit		
0796	59	str_scan_false:	pop	cx	;restore length of the string
0797	5E		pop	si	; and its start
0798	F9		stc		;set failure flag
0799	5B	str_scan_xit:	pop	bx	
079A	C3		ret		
079B		str_scan	endp		

```
        page
;
; str_name      Scan a string for a name terminated with blank or comma.
;                Discard leading blanks or commas.
;                Abort (with stack bad!) to error_name if it has a bad
;                character.
;
;                ds:si is the start of the string
;                cx    is the length of the string
;
;                Return NC if name found, length in BX.
;                Return C  if name not found.
;                si/cx is unchanged, except for leading blank/comma discard.
;                Destroys ax.
;

079B      str_name      proc    near

079B  E3 2E          str_name_lead: jcxz   str_name_none
079D  E8 076E R       call    str_skip_blanks      ;skip leading blanks
07A0  E3 29          jcxz   str_name_none
07A2  80 3C 2C        cmp    byte ptr ds:[si],',' ;and commas
07A5  75 04          jne    str_name_name
07A7  46             inc    si
07A8  49             dec    cx
07A9  EB F0          jmp    short str_name_lead

07AB  56             str_name_name: push   si           ;save ptr to start
07AC  51             push   cx           ;save length
07AD  46             str_name_name1: inc    si
07AE  49             dec    cx
07AF  E3 12          jcxz   str_name_end
07B1  8A 04          mov    al,byte ptr ds:[si]
07B3  E8 07CD R       call    chk_bad_char     ;check for unprintables
07B8  E9 052A R       $if    e
                           jmp    error_name
$endif
07BB  3C 2C          cmp    al,','
07BD  74 04          je     str_name_end      ;scan for comma
07BF  3C 20          cmp    al,' '
07C1  75 EA          jne    str_name_name1  ;or blank

07C3  8B DE          str_name_end:  mov    bx,si      ;compute length in bx
07C5  59             pop    cx           ;and restore ptr/length
07C6  5E             pop    si
07C7  2B DE          sub    bx,si
07C9  F8             clc
07CA  C3             ret

07CB  F9             str_name_none: stc
07CC  C3             ret           ;return C if no name

07CD      str_name      endp
```

```

; chk_bad_char    Check if the char (al) is bad, ie not printable.
;                   We are quite liberal in this, and allow all sorts of
;                   weird characters except those that cause cursor motion.
;
;                   Return z if bad.  Destroys nothing.
;

07CD      chk_bad_char    proc    near
07CD      3C 0D          cmp     al,cr
07CF      74 0E          je      chk_bad_z
07D1      3C 08          cmp     al,bs
07D3      74 0A          je      chk_bad_z
07D5      3C 07          cmp     al,bell
07D7      74 06          je      chk_bad_z
07D9      3C 0A          cmp     al,Tf
07DB      74 02          je      chk_bad_z
07DD      3C 00          cmp     al,_
07DF      C3             ret
07E0      chk_bad_z:    endp

```

```
page

;(This is a modified version of cwp's routine from his string package)

;
; str_to_num: Convert the front of a string to a number. Does NOT change
; string at all. This routine handles decimal, and hex if the number is
; prefixed by "$". Any non-digit stops the conversion. Blanks following
; the number are skipped.
;
; entry:
;   si:string
;   cx:length of string
; exit:
;   ax:number
;   si:remaining part of string
;   cx:length of remaining part of the string
;   cy:on if bad number
;

07E0      str_to_num    proc    near
07E0      53          push    bx
07E1      52          push    dx
07E2      57          push    di

07E3      B8 0000      mov     ax,0      ;clear total
07E6      B7 00          mov     bh,0      ;use for byte to word
07E8      BA 000A      mov     dx,10     ;default base is decimal
07EB      BB FE          mov     di,si     ;keep a pointer to first char
07ED      E3 58          jcxz   str_num_err ;exit if string empty
07EF      80 3C 24      cmp     byte ptr [si],'$'    ;hex prefix ?
07F2      75 07          jne    no_prefix
07F4      BA 0010      mov     dx,16     ;base is hex
07F7      46          inc    si        ;move past prefix
07F8      49          dec    cx
07F9      E3 4C          jcxz   str_num_err

07FB      no_prefix:    s_t_n_loop:
07FB      8A 1C          mov     bl,[si]    ;get a char
07FD      80 FB 30      cmp     bl,'0'     ;check for dec. digit
0800      7C 33          jl     delim_check ;exit on non-digit
0802      80 FB 39      cmp     bl,'9'
0805      7E 22          jle    dec_digit
0807      80 FB 61      cmp     bl,'a'     ;check for lower case
080A      7C 08          jl     not_lower
080C      80 FB 66      cmp     bl,'f'
080F      7F 24          jg     delim_check ;nothing above 'f'
0811      80 EB 20      sub    bl,'a'-'A' ;convert to upper
0814      not_lower:    s_t_n_loop:
0814      80 FB 41      cmp     bl,'A'     ;check for hex digit
0817      7C 1C          jl     delim_check
0819      80 FB 46      cmp     bl,'F'
081C      7F 17          jg     delim_check
081E      83 FA 10      cmp     dx,16     ;hex digit, are we in hex mode ?
```

```
0821 75 12          jne    delim_check      ;exit if wrong mode
0823 80 EB 37        sub    bl,'A'-'0'       ;convert hex to num.
0826 EB 04 90        jmp    s_t_n_add

0829          ;dec_digit:
0829 80 EB 30        sub    b1,'0'          ;convert dec. to num.
082C          s_t_n_add:
082C 52             push   dx              ;save base
082D F7 E2           mul    dx
082F 5A             pop    dx
0830 03 C3           add    ax,bx
0832 46             inc    si              ;move to next char.
0833 E2 C6           loop   s_t_n_loop     ;do for all digits

0835          delim_check:
0835          str_num_exit:
0835 8B DE           mov    bx,si
0837 2B DF           sub    bx,di          ;how many chars scanned
0839 83 FB 01         cmp    bx,1           ;two or more digits is fine
083C 7F 13           jg    str_num_ok
083E 0B DB           or    bx,bx          ;zero digits is no good
0840 74 05           jz    str_num_err
0842 83 FA 0A         cmp    dx,10          ;if base was decimal, one char. is ok
0845 74 0A           je    str_num_ok

0847          str_num_err:
0847 2B F7           sub    si,di          ;---Failure exit---
0849 03 CE           add    cx,si          ;# of bytes we scanned
084B 8B F7           mov    si,di          ;restore length of string
084D F9             stc
084E EB 05 90         jmp    str_num_exit_2  ;set fail flag

0851          ;
0851 E8 076E R        str_num_ok:
0854 F8             call   str_skip_blanks ;skip following blanks
0855          str_num_exit_2:
0855 5F             pop    di
0856 5A             pop    dx
0857 5B             pop    bx
0858 C3             ret
0859          str_to_num:
0859          endp

0859          cseg
0859          ends
0859          end
```

Macros:

	Name	Length
\$DO.	.	000E
\$DOJCXZ.	.	0001
\$DOJMP.	.	0001
\$DOLOOP.	.	0001
\$DOUNTIL.	.	0002
\$DOWHILE.	.	0002
\$ELSE.	.	0006
\$ELSEIF.	.	0008
\$ELSEIFNOT	.	0008
\$ENDIF.	.	0009
\$EXITIF.	.	0004
\$GETN.	.	0001
\$GETT.	.	0001
\$IF.	.	0006
\$IFNOT	.	0006
\$JMP.	.	0001
\$LAB.	.	0001
\$PUTN.	.	0001
\$PUTT.	.	0001
\$REPEAT.	.	0007
\$REPEATLOOP.	.	0007
\$REPEATUNTIL.	.	0007
\$REPEATWHILE	.	0007
L4_CALL.	.	0003
VERSION.	.	0002

Structures and records:

Name	Width	# fields		
	Shift	Width	Mask	Initial
ETHER_HEADER	002E	0015		
ARC_CODE		0000		
GARBAGE		0001		
PACKET_NUM		0002		
FRAGMENT		0003		
CHECKSUM		0004		
E_LENGTH		0006		
TRANS_CTRL		0008		
PACKET_TYPE		0009		
DEST_NETWORK		000A		
DEST_HOST		000E		
DEST_SOCKET		0014		
SRC_NETWORK		0016		
SRC_HOST		001A		
SRC_SOCKET		0020		
CONN_CTRL		0022		
DATA_TYPE		0023		
SOURCE_ID		0024		
DEST_ID		0026		
SEQ_NUM		0028		

ACK_NUM.	002A
ALLOC_NUM.	002C
GRP_REC.	0003
GRP_NAMEL.	0000
GRP_NAME.	0001
HOST_ID.	0006
LEVEL_4_PUBS.	004B
L4_VERSION.	0000
L4_FEATURES.	0002
L4_STATUS.	0004
OUR_ARC.	0005
L4_DEBUG.	0006
L4_IN_USE.	0008
LONG_PKT_MODE.	0009
OUR_ETHER.	000A
FREE_HEAD.	0010
ACTIVE_HEAD.	0014
D_TO_ACCEPT_WAIT.	0018
D_CONN_TRIES.	001A
D_TO_ACK_WAIT.	001C
D_MESSAGE_TRIES.	001E
D_TO_PKT_WAIT.	0020
TO_PKT_KEEP.	0022
TO_TA_WAIT.	0024
OLD_DISKIO_VECTOR.	0026
ABORT_VECTOR.	002A
USER_EXIT_VECTOR.	002E
NIC_SEG.	0032
PUB_SPARE1.	0034
PUB_SPARE2.	0036
PUB_SPARE3.	0038
PUB_SPARE4.	003A
OLD_BOOT_VECTOR.	003C
BOOT_HOST.	0040
BOOT_RB.	0046
BOOTED_FROM_NET.	004A
REQ_BLOCK.	0086
NEXT.	0043
RB_SIG.	0000
PROTOCOL_MODE.	0004
RB_IN_USE.	0006
CONN_STATUS.	0007
SEND_STATUS.	0008
SEND_PTR.	0009
SEND_LENGTH.	000A
HDR_ARC_CODE.	000E
HDR_GARBAGE.	0010
HDR_PACKET_NUM.	0011
HDR_FRAGMENT.	0012
HDR_CHECKSUM.	0013
HDR_E_LENGTH.	0014
HDR_TRANS_CTRL.	0016
HDR_PACKET_TYPE.	0018
HDR_DEST_NETWORK.	0019

HDR_DEST_HOST	001E
HDR_DEST_SOCKET	0024
HDR_SRC_NETWORK	0026
HDR_SRC_HOST	002A
HDR_SRC_SOCKET	0030
HDR_CONN_CTRL	0032
HDR_DATA_TYPE	0033
HDR_SOURCE_ID	0034
HDR_DEST_ID	0036
HDR_SEQ_NUM	0038
HDR_ACK_NUM	003A
HDR_ALLOC_NUM	003C
RECV_STATUS	003E
RECV_TYPE	003F
RECV_PTR	0040
RECV_LIMIT	0044
RECV_LENGTH	0046
RB_TO_ACCEPT_WAIT	0048
RB_CONN_TRIES	004A
RB_TO_ACK_WAIT	004C
RB_MESSAGE_TRIES	004E
RB_TO_PKT_WAIT	0050
PEND_VALID	0052
PEND_TYPE	0053
HIS_ARC	0054
HIS_BCST	0055
RB_SPARE1	0056
RB_SPARE2	0058
RB_SPARE3	005A
RB_SPARE4	005C
CONN_STATE	005E
SEND_STATE	005F
SEND_CHANGED	0060
SEND_CURSOR	0062
SEND_REMAINING	0066
OUR_ACK_REQ	0068
OUR_REQ_VALID	006A
RECV_STATE	006B
RECV_CHANGED	006C
RECV_CURSOR	006E
HIS_SEQ	0072
HIS_ACK	0074
HIS_ALLOC	0076
PEND_BUF	0078
PEND_HDR	007A
PEND_START	007E
SEND_RETRIES	0080
MESS_START_SEQ	0082
ACK_FLAGS	0084
L2_SEQ	0085
STN_REC	0007
STN_ADDR	0000
STN_FLAGS	0001
STN_GROUPMASK	0002

```
STN_NAMEL. . . . . 0004
STN_NAME . . . . . 0005
TIME_OUT_REC . . . . . 000A 0005
TO_ACCEPT_WAIT . . . . . 0000
CONN_TRIES . . . . . 0002
TO_ACK_WAIT. . . . . 0004
MESSAGE_TRIES. . . . . 0006
TO_PKT_WAIT. . . . . 0008
```

Segments and Groups:

Name	Size	Align	Combine	Class
CGRP GROUP				
HSEG	0000	PARA	COMMON	'HSEGCL'
VSEG	3953	PARA	COMMON	'VSEGCL'
CSEG	0859	PARA	PUBLIC	'CODE'

Symbols:

Name	Type	Value	Attr
ADDR_ALLNumber	FFFF	
AL4LOCATE.L NEAR	0000	CSEG External
ASSERTAlias	C TRUE	
AUTO_POPDOWNL BYTE	002D	VSEG
AUTO_POPUPL BYTE	002B	VSEG
BAD_NIC.Number	0002	
BAD_RAM.Number	0004	
BAD_RIM.Number	0003	
BAD_SOCKNumber	0004	
BELLNumber	0007	
BORDERS.L BYTE	0494	CSEG
BRDCST_ARCNET.L BYTE	0C58	VSEG
BSNumber	0008	
CHANNEL.L BYTE	0030	VSEG
CHANNEL_NAMEL BYTE	0504	CSEG
CHAR_BOLD.L BYTE	0033	VSEG
CHAR_BOLDLINE.L BYTE	0035	VSEG
CHAR_LINE.L BYTE	0034	VSEG
CHAR_NORMAL.L BYTE	0032	VSEG
CHK_BAD_CHARN PROC	07CD	CSEG Length =0013
CHK_BAD_Z.L NEAR	07DF	CSEG
CH_GRPNumber	0004	
CH_GRP_ATRNumber	0008	
CH_STNNumber	0001	
CH_STN_ATRNumber	0002	
CN_ACCEPT_WAITNumber	0005	
CN_ESTABLISHEDNumber	0000	
CN_FAIL.Number	0002	
CN_NOT_CONN.Number	0001	
CN_OPEN_RECVNumber	0006	
CN_PARM_ERROR.Number	0004	
CN_STATE_ERRORNumber	0003	

COLOR.L BYTE	0031	VSEG
CRNumber	000D	
CURTASK.L WORD	000F	VSEG
C_FALSE.Number	0000	
C_TRUE.Number	FFFF	
DÉC_DIGIT.L NEAR	0829	CSEG
DELIM_CHECK.L NEAR	0835	CSEG
DEST_GRPL BYTE	0C4F	VSEG
DEST_STNL BYTE	0C50	VSEG
DISPLAY_BORDERL BYTE	04B8	CSEG
DOSINT_EXIT.Number	004C	
DOSINT_EXITSTAY.Number	0031	
DOSINT_PRINTC.Number	0002	
DOSINT_PRINTS.Number	0009	
DOS_INT.Number	0021	
DO_OPEN.L BYTE	0023	VSEG
DO_STOP.L BYTE	0024	VSEG
DWIN_BOTTOM_ROW.L BYTE	0048	VSEG
DWIN_LEFT_COL.L BYTE	0045	VSEG
DWIN_LOWERRIGHT.L WORD	0047	VSEG
DWIN_RIGHT_COL.L BYTE	0047	VSEG
DWIN_TOP_ROW.L BYTE	0046	VSEG
DWIN_UPPERLEFT.L WORD	0045	VSEG
EQUIP_INT.Number	0011	
ERROR_CHANNEL.L NEAR	0534	CSEG
ERROR_COLOR.L NEAR	0539	CSEG
ERROR_INSTALL.L NEAR	0525	CSEG
ERROR_KEY.L NEAR	051B	CSEG
ERROR_MSG.L NEAR	0543	CSEG
ERROR_NAME.L NEAR	052A	CSEG
ERROR_NWINIT.L NEAR	050C	CSEG
ERROR_PARMS.L NEAR	0520	CSEG
ERROR_POPDOWN.L NEAR	052F	CSEG
ERROR_RB.L NEAR	0511	CSEG
ERROR_SOCKET.L NEAR	0516	CSEG
ERROR_USERWIND.L NEAR	053E	CSEG
ESC.Number	001B	
FALSE.Number	0000	
FIRST_DISPATCH.L BYTE	0025	VSEG
FROM_TO_SIZE.Number	0014	
GENL_MSG_TYPE.Number	0099	
GET_ENV_FOUND.L NEAR	0744	CSEG
GET_ENV_STRING.N PROC	071B	CSEG
GET_RB.N PROC	06DA	CSEG
GOOD_RB_SIG.Number	6272	
GROUP_BORDER.L BYTE	0494	CSEG
GROUP_WINDOW.L BYTE	160F	VSEG
GRP_COUNT.L BYTE	0B70	VSEG
GRP_LIST.L BYTE	0B71	VSEG
GRP_LIST_SIZE.L WORD	0B6E	VSEG
GRP_NUMBER_MAP.L BYTE	0C3E	VSEG
GRP_REC_FSIZE.Number	0001	
GRP_SUBTYPE.Number	00BB	
GWIN_BOTTOM_ROW.L BYTE	0040	VSEG

GWIN_LEFT_COL.L BYTE	003D	VSEG
GWIN_LOWERRIGHT.L WORD	003F	VSEG
GWIN_RIGHT_COL.L BYTE	003F	VSEG
GWIN_TOP_ROW.L BYTE	003E	VSEG
GWIN_UPPERLEFT.L WORD	003D	VSEG
HOW_OFTEEN.Number	0003	
IF\$1002.L NEAR	013B	CSEG
IF\$102.L NEAR	0039	CSEG
IF\$1052.L NEAR	015B	CSEG
IF\$1100.L NEAR	0180	CSEG
IF\$1150.L NEAR	0221	CSEG
IF\$1202.L NEAR	0252	CSEG
IF\$1250.L NEAR	0248	CSEG
IF\$1302.L NEAR	0262	CSEG
IF\$1352.L NEAR	02AC	CSEG
IF\$1400.L NEAR	0340	CSEG
IF\$1402.L NEAR	030D	CSEG
IF\$1450.L NEAR	02D7	CSEG
IF\$1451.L NEAR	02EF	CSEG
IF\$1502.L NEAR	02E6	CSEG
IF\$152.L NEAR	0049	CSEG
IF\$1552.L NEAR	02FE	CSEG
IF\$1602.L NEAR	03A0	CSEG
IF\$1650.L NEAR	035B	CSEG
IF\$1651.L NEAR	03A0	CSEG
IF\$1702.L NEAR	0368	CSEG
IF\$1752.L NEAR	0433	CSEG
IF\$1800.L NEAR	041B	CSEG
IF\$1852.L NEAR	042B	CSEG
IF\$1902.L NEAR	06E8	CSEG
IF\$1950.L NEAR	0710	CSEG
IF\$2000.L NEAR	0727	CSEG
IF\$202.L NEAR	0059	CSEG
IF\$2050.L NEAR	0732	CSEG
IF\$2051.L NEAR	073B	CSEG
IF\$2100.L NEAR	0746	CSEG
IF\$2101.L NEAR	074F	CSEG
IF\$2152.L NEAR	07BB	CSEG
IF\$252.L NEAR	009E	CSEG
IF\$300.L NEAR	009C	CSEG
IF\$302.L NEAR	008D	CSEG
IF\$352.L NEAR	0073	CSEG
IF\$402.L NEAR	009C	CSEG
IF\$452.L NEAR	00AF	CSEG
IF\$502.L NEAR	00D4	CSEG
IF\$52.L NEAR	0018	CSEG
IF\$552.L NEAR	00D1	CSEG
IF\$602.L NEAR	00CA	CSEG
IF\$652.L NEAR	00F9	CSEG
IF\$702.L NEAR	00E5	CSEG
IF\$752.L NEAR	00ED	CSEG
IF\$802.L NEAR	0122	CSEG
IF\$852.L NEAR	010A	CSEG
IF\$902.L NEAR	0151	CSEG

IF\$952L NEAR	0133	CSEG
IF\$LNumber	0000	
IF\$NNumber	0866	
IF\$NSNumber	0868	
IF\$NS1Number	0866	
IF\$NS2Number	0802	
IF\$NS3Number	073A	
IF\$TNumber	0002	
IF\$T1Number	0002	
IF\$T2Number	0001	
IF\$T3Number	0002	
INFINITYNumber	FFFF	
INIT_TASK.N PROC	0702	CSEG
INSTALL_OK_MSGL BYTE	04DC	CSEG
IS_DEST_STN.L BYTE	0C51	VSEG
J_ERROR_KEY.L NEAR	0070	CSEG
J_ERROR_USERWIND	.L NEAR	0107	CSEG
K18.L WORD	0492	CSEG
KB10L BYTE	0C9F	VSEG
KBCHAR_MSG_TYPE.Number	0094	
KBD_BUFFERL BYTE	005F	VSEG
KBD_BUFFER_FULL.L BYTE	0054	VSEG
KBD_BUFFER_PREV.L BYTE	00EE	VSEG
KBD_BUFSIZE.Number	008F	
KBD_PREVBUFPTRL WORD	0905	VSEG
KBD_PREVBUFSIZL WORD	0903	VSEG
KBD_PREV_INSL BYTE	0907	VSEG
KB_CTRL.Number	0061	
KB_CTRL_SPKRNumber	0003	
KB_CURSOR.L BYTE	0055	VSEG
KB_INT_BUSY.L BYTE	0020	VSEG
KB_TASK.L NEAR	0000	CSEG
KEYCHAR.L BYTE	005D	VSEG
KEYCHAR_SCANL BYTE	005E	VSEG
KEY_DEL.Number	5300	
KEY_DOWN.Number	5000	
KEY_END.Number	4F00	
KEY_EXT_ACTIVE.Number	0001	
KEY_EXT_IDLE.Number	0002	
KEY_F1Number	3800	
KEY_F3Number	3D00	
KEY_HOME.Number	4700	
KEY_INS.Number	5200	
KEY_INT.Number	0016	
KEY_INT_RTN.L FAR	0000	CSEG
KEY_LEFT.Number	4B00	
KEY_NWCMDSNumber	0062	
KEY_PGDNNumber	5100	
KEY_PGUPNumber	4900	
KEY_READNumber	0000	
KEY_RIGHT.Number	4D00	
KEY_SHIFT.Number	0002	
KEY_STATNumber	0001	
KEY_TALKNumber	0063	

KEY_TALK_SEND.Number	0061		
KEY_TALK_START.Number	005F		
KEY_TALK_STOP.Number	0060		
KEY_UP.Number	4800		
KW1092.L WORD	0CA0	VSEG	
L4_ENTRIES.V DWORD	0000	CSEG	External
L4_EXIT_BUSY.L BYTE	001F	VSEG	
L4_EXIT_RTN.L FAR	0000	CSEG	External
L4_IN_OUR_SEG.Number	0000		
L4_OK.Number	0000		
L4_PUBLICS.L DWORD	0C66	VSEG	
LAST_OPEN_RCV.L BYTE	0C5B	VSEG	
LF.Number	000A		
LISTS_CHGD.L BYTE	0911	VSEG	
LOGOFF_MSG_TYPE.Number	0097		
LOGON_MSG_TYPE.Number	0098		
L_CHANNEL_NAME.Number	0008		
L_TALK_NAME.Number	0009		
MAX_ENTRY.Alias	OFF_L4GET_PTRS		
MAX_GROUPS.Number	0010		
MAX_GRP_LIST.Number	00C8		
MAX_NAMESIZE.Number	0010		
MAX_STN_LIST.Number	0258		
MSG_CHANNEL.L BYTE	0673	CSEG	
MSG_COLOR.L BYTE	0695	CSEG	
MSG_INSTALL.L BYTE	05E5	CSEG	
MSG_KEY.L BYTE	059B	CSEG	
MSG_NAME.L BYTE	05FF	CSEG	
MSG_NWINIT.L BYTE	0553	CSEG	
MSG_PARMS.L BYTE	05B9	CSEG	
MSG_POPDOWN.L BYTE	064A	CSEG	
MSG_RB.L BYTE	056B	CSEG	
MSG_SOCKET.L BYTE	0585	CSEG	
MSG_USERWIND.L BYTE	06B5	CSEG	
NIL.Number	0000		
NOT_LOWER.L NEAR	0814	CSEG	
NO_L4.Number	0005		
NO_NIC.Number	0001		
NO_PREFIX.L NEAR	07FB	CSEG	
NTASKS.Number	0004		
N_PAGES.Number	0003		
OFF_ABORT_CONN.Number	0009		
OFF_ACK_NOW.Number	0006		
OFF_ACTIVATE_RB.Number	0001		
OFF_CONNECT.Number	0007		
OFF_DISCONNECT.Number	0008		
OFF_IGNORE.Number	0004		
OFF_L4CHURN.Number	000C		
OFF_L4GET_PTRS.Number	000D		
OFF_L4INIT.Number	0000		
OFF_LISTEN.Number	0003		
OFF_OPEN_RECV.Number	0005		
OFF_RECV_MSG.Number	000B		
OFF_RELEASE_RB.Number	0002		

OFF_SEND_MSGNumber	000A	
OLD_KEY_VECTORL DWORD	0C75	VSEG
OLD_L4_EXITL DWORD	0C6C	VSEG
OLD_TIME_VECL DWORD	0C79	VSEG
ONE_SECONDNumber	0012	
ON_NICNumber	0000	
OPEN_RECVDL BYTE	0026	VSEG
OTHER_CONN_BUSYL BYTE	0021	VSEG
OUR_ARCNETL BYTE	0C59	VSEG
OUR_L4_DEBUGL WORD	0C6A	VSEG
PARMS_DONEL NEAR	015B	CSEG
PARM_LENV BYTE	0000	CSEG
PARM_STRV BYTE	0000	CSEG
POPDOWN_TIMEL WORD	002E	VSEG
POPPED_UPL BYTE	0029	VSEG
PSPL WORD	0C73	VSEG
PSP_ENVIRONNumber	002C	
QCHANNELL BYTE	0479	CSEG
QCOLORL BYTE	0481	CSEG
QDEBUGL BYTE	045A	CSEG
QFL BYTE	0469	CSEG
QGROUPL BYTE	0449	CSEG
QKEYL BYTE	0465	CSEG
QPOPDOWNL BYTE	0471	CSEG
QPOPUPL BYTE	046B	CSEG
QTESTL BYTE	0460	CSEG
QTIMESTAMPL BYTE	0450	CSEG
QUSERL BYTE	0443	CSEG
QUSERWINDOWL BYTE	0487	CSEG
RCV_ARCNETL BYTE	0C56	VSEG
RCV_BUFFERL BYTE	017D	VSEG
RCV_BUFSIZENumber	03C3	
RCV_MSG_TYPEL BYTE	0C5A	VSEG
RCV_RBL DWORD	0C5E	VSEG
RCV_TASKL NEAR	0000	CSEG
RECV_BUSYL BYTE	001D	VSEG
RELEASE_RBSL NEAR	0000	CSEG
RETRIESNumber	0002	
RQSCRN_MSG_TYPENumber	0096	
RUMORNumber	0080	
RUMOR_COUNTL BYTE	0915	VSEG
SAVSPL WORD	0013	VSEG
SAVSSL WORD	0011	VSEG
SCAN_KEYWORDSL NEAR	0028	CSEG
SCRN_MSG_TYPENumber	0095	
SELECT_DWINNumber	0601	
SELECT_GWINNumber	0604	
SELECT_SWINNumber	0603	
SELECT_TWINNumber	0602	
SEND_BUSYL BYTE	001C	VSEG
SEND_MSG_PTRL DWORD	0057	VSEG
SEND_MSG_RQSTL BYTE	0027	VSEG
SEND_MSG_STATL BYTE	005C	VSEG
SEND_MSG_STNL BYTE	005B	VSEG

SEND_NG_COUNT.	L BYTE	0C53	VSEG	
SEND_OK_COUNT.	L BYTE	0C52	VSEG	
SEND_TASK.	L NEAR	0000	CSEG	External
SHOW_RUMORS.	Alias	C_TRUE		
SND_ARCNET.	L BYTE	0C57	VSEG	
SND_BUFFER.	L BYTE	0540	VSEG	Length =03C3
SND_BUFFER_SIZE.	L WORD	0908	VSEG	
SND_BUFFER_TXT.	L WORD	090A	VSEG	
SND_BUFSIZE.	Number	03C3		
SND_RB.	L DWORD	0C62	VSEG	
SOCK_IN_USE.	Number	0003		
SOCK_NOT_FOUND.	Number	0001		
SOCK_OK.	Number	0000		
STACK0.	L WORD	0F00	VSEG	
STACK1.	L WORD	1158	VSEG	
STACK2.	L WORD	13B0	VSEG	
STACK3.	L WORD	1608	VSEG	
STATION_BORDER.	L BYTE	04A6	CSEG	
STN_COUNT.	L BYTE	0914	VSEG	
STN_LIST.	L BYTE	0916	VSEG	Length =0258
STN_LIST_SIZE.	L WORD	0912	VSEG	
STN_REC_FSIZE.	Number	0005		
STN_SUBTYPE.	Number	00AA		
STR_NAME.	N PROC	079B	CSEG	Length =0032
STR_NAME_END.	L NEAR	07C3	CSEG	
STR_NAME_LEAD.	L NEAR	079B	CSEG	
STR_NAME_NAME.	L NEAR	07AB	CSEG	
STR_NAME_NAMEL.	L NEAR	07AD	CSEG	
STR_NAME_NONE.	L NEAR	07CB	CSEG	
STR_NUM_ERR.	L NEAR	0847	CSEG	
STR_NUM_EXIT.	L NEAR	0835	CSEG	
STR_NUM_EXIT_2.	L NEAR	0855	CSEG	
STR_NUM_OK.	L NEAR	0851	CSEG	
STR_SCAN.	N PROC	0779	CSEG	Length =0022
STR_SCAN_FALSE.	L NEAR	0796	CSEG	
STR_SCAN_XIT.	L NEAR	0799	CSEG	
STR_SKIP_BLANKS.	N PROC	076E	CSEG	Length =0008
STR_SKIP_LOOP.	L NEAR	0770	CSEG	
STR_SKIP_RET.	L NEAR	0778	CSEG	
STR_TO_NUM.	N PROC	07E0	CSEG	Length =0079
STR_UPPER.	N PROC	0757	CSEG	Length =0017
STR_UPPER_LOOP.	L NEAR	075B	CSEG	
STR_UPPER_NXT.	L NEAR	0769	CSEG	
STR_UPPER_XIT.	L NEAR	076D	CSEG	
SUSPENDED.	L BYTE	0028	VSEG	
SWIN_BOTTOM_ROW.	L BYTE	0044	VSEG	
SWIN_LEFT_COL.	L BYTE	0041	VSEG	
SWIN_LOWERRIGHT.	L WORD	0043	VSEG	
SWIN_RIGHT_COL.	L BYTE	0043	VSEG	
SWIN_TOP_ROW.	L BYTE	0042	VSEG	
SWIN_UPPERLEFT.	L WORD	0041	VSEG	
S_T_N_ADD.	L NEAR	082C	CSEG	
S_T_N_LOOP.	L NEAR	07FB	CSEG	
TALKINIT.	N PROC	0000	CSEG	Global Length =06DA

TALK_EXIT.L NEAR	0000	CSEG	External
TALK_NAME.L BYTE	04FB	CSEG	
TALK_WKS.Number	0D04		
TASK0.L WORD	0007	VSEG	
TASK1.L WORD	0009	VSEG	
TASK2.L WORD	000B	VSEG	
TASK3.L WORD	000D	VSEG	
TASKS_ACTIVE.L BYTE	001B	VSEG	
TCBS.L WORD	0007	VSEG	
TCBSIZE.Number	0002		
TEST_MODE.L BYTE	002A	VSEG	
TF_ACK_WAIT.Number	0006		
TF_FAIL.Number	0002		
TF_IDLE.Number	0000		
TF_IN_PROG.Number	0005		
TF_NOT_CONN.Number	0001		
TF_NOT_IMPL.Number	0008		
TF_PARM_ERROR.Number	0004		
TF_RECV_OVFL.Number	0007		
TF_STATE_ERROR.Number	0003		
TICKS.L WORD	0C7D	VSEG	
TIMEOUT.Number	0006		
TIMER_CH2.Number	0042		
TIMER_CTRL.Number	0043		
TIMER_CTRL_MSB.Number	00A6		
TIMER_INT.Number	001A		
TIMER_INT_BUSY.L BYTE	001E	VSEG	
TIMER_INT_RTN.L FAR	0000	CSEG	External
TIMER_READ.Number	0000		
TIMER_TICK.Number	0008		
TIMESTAMP.L BYTE	002C	VSEG	
TO_MANY_SOCKS.Number	0002		
TO_HEX.L NEAR	0000	CSEG	External
TRUE.Number	0001		
TWEEDLE1.L BYTE	0C81	VSEG	
TWEEDLE2.L BYTE	0C85	VSEG	
TWEEDLE3.L BYTE	0C89	VSEG	
TWEEDLE4.L BYTE	0C8B	VSEG	
TWEEDLE5.L BYTE	0C8F	VSEG	
TWEEDLE6.L BYTE	0C92	VSEG	
TWEEDLE_PTR.L WORD	0C7F	VSEG	
TWIN_BOTTOM_ROW.L BYTE	004C	VSEG	
TWIN_LEFT_COL.L BYTE	0049	VSEG	
TWIN_LOWERRIGHT.L WORD	004B	VSEG	
TWIN_RIGHT_COL.L BYTE	004B	VSEG	
TWIN_TOP_ROW.L BYTE	004A	VSEG	
TWIN_UPPERLEFT.L WORD	0049	VSEG	
TXT_SUBTYPE.Number	00CC		
TYING_BORDER.L BYTE	04CA	CSEG	
VIDEO_GETCURSOR.Number	0003		
VIDEO_INT.Number	0010		
VIDEO_SCROLL.Number	0006		
VIDEO_SETCURSOR.Number	0002		
VIDEO_WRITE_CH.Number	000A		

VIDEO_WRITE_TTY.Number	000E	
WINDOWL NEAR	0000	
WINDOWS_OPENL BYTE	0022	VSEG
WINDOW_BUFFERSL BYTE	160F	VSEG
WINDOW_CLOSENumber	0002	
WINDOW_DEFINENumber	0000	
WINDOW_GETCURSNumber	0004	
WINDOW_INITNumber	0007	
WINDOW_OPENNumber	0001	
WINDOW_PAGEOPNumber	0009	
WINDOW_PRINTNumber	0003	
WINDOW_SELECTNumber	0006	
WINDOW_SETATTRNumber	0008	
WINDOW_SETCURSNumber	0005	
WKS.L WORD	0C5C	VSEG
XNS_ECHONumber	0003	
XNS_ERRORNumber	0002	
XNS_NONENumber	0000	
XNS_PEPNumber	0005	
XNS_RIPNumber	0001	
XNS_SPPNumber	0004	

13206 Bytes free

Warning Severe
Errors Errors
0 0