```
1   ************************************************************
2   *                                                          *
3   *                A T T A C H . S L A V E                   *
4   *                                                          *
5   *            Michael J. Mahon - Sep 29, 2004               *
6   *                 Revised Apr 30, 2010                     *
7   *                                                          *
8   *      Copyright (c) 2004, 2007, 2008, 2009, 2010          *
9   *                                                          *
10  *   ATTACH.SLAVE is the slave server for ATTACH, which     *
11  *   allows another machine on the network to control the   *
12  *   slave machine by substituting for its keyboard and     *
13  *   display.                                               *
14  *                                                          *
15  *   The install code uses the current 'CSW' address for    *
16  *   local echo if DOS or ProDOS is active, then sets up    *
17  *   the character input and output vectors redirecting     *
18  *   all output to a circular buffer in the master machine  *
19  *   and taking all input from a local circular buffer.     *
20  *                                                          *
21  ************************************************************
22  *                                                          *
23  *                     Change History                       *
24  *                                                          *
25  *   04/30/10:                                              *
26  *                                                          *
27  *   Acquire lock on CB using new PEEKPOKE instead of       *
28  *   PEEKINC, since "set to 1" is more reliable.            *
29  *                                                          *
30  *   01/29/09:                                              *
31  *                                                          *
32  *   Fixed deadlock w/ master when the slave sends while    *
33  *   the master is sending an input line, set "timeout"     *
34  *   to 1 (only 1 x 3 retries, or about 60ms.) and added    *
35  *   40 ms. "serve" calls to slave release retries.         *
36  *                                                          *
37  *   Added mechanism to allow ATTACH to detect when slave   *
38  *   is already attached by another machine.  The first     *
39  *   two bytes of ATTACH.SLAVE are a "signature" of being   *
40  *   attached, clobbered by DETACH, and the third byte is   *
41  *   the ID of the attaching machine.                       *
42  *                                                          *
43  *   01/03/09:                                              *
44  *                                                          *
45  *   Modified initialization to check for ProDOS and DOS,   *
46  *   then handle other cases (CRATE, MSERVE) as "bare".     *
47  *                                                          *
48  *   12/19/08:                                              *
49  *                                                          *
50  *   Removed output "prefix" code, since ATTACH can only    *
51  *   be attached to one machine at a time.                  *
52  *                                                          *
53  *   12/10/08:                                              *
54  *                                                          *
```

```
55   *   Suppressed echo of CR (just like all other chars).   *
56   *   Deleted ctl-Z detach mechanism from Keyin.           *
57   *                                                        *
58   *   12/06/08:                                            *
59   *                                                        *
60   *   Added DETACH entry point.  Changed output prefix to  *
61   *   3 characters: <2-dig decimal ID> + ":".              *
62   *                                                        *
63   *   11/25/08:                                            *
64   *                                                        *
65   *   Added code to save CSW/KSW when ATTACHing and to     *
66   *   restore them when DETACHing, to keep any active      *
67   *   OS connected after the machine is DETACHed.          *
68   *                                                        *
69   *   Moved 'ATTACH' to end so that it is overwritten by   *
70   *   outbuf and inbuf.  Special-cased one reloc in ATTACH.*
71   *                                                        *
72   *   07/11/07:                                            *
73   *                                                        *
74   *   Re-wrote communication routines to use circular      *
75   *   buffers for passing messages directly instead of     *
76   *   using a Message Server.                              *
77   *                                                        *
78   *   Expanded simple relocator to handle up to 512 byte   *
79   *   range of code to be relocated.                       *
80   *                                                        *
81   *   12/14/04:                                            *
82   *                                                        *
83   *   Added ctl-Z command to "detach" by restoring saved   *
84   *   I/O hooks, then giving control back to 'warmstrt'.   *
85   *                                                        *
86   *   Changed install routine (ATTACH) to be OS-specific   *
87   *   and to save the current I/O hooks, and then install  *
88   *   new output and input hooks to keep OS connected.     *
89   *                                                        *
90   *   11/13/04:                                            *
91   *                                                        *
92   *   Changed load address to $6000.                       *
93   *                                                        *
94   *   Now gets input and output queue numbers from param   *
95   *   area just after entry JMP.  Allows ATTACH to set up  *
96   *   queues.                                              *
97   *                                                        *
98   *   11/08/04:                                            *
99   *                                                        *
100  *   Removed suppression of CR echo to fix output bug.    *
101  *                                                        *
102  *   Moved initialization of 'sbuf+dst' and 'sbuf+len+1'  *
103  *   from ATTACH to just prior to PUTMSG call, so that    *
104  *   ATTACH.SLAVE can be used to perform network requests *
105  *   (which change the contents of 'sbuf').               *
106  *                                                        *
```

```
107  *   11/05/04:                                                  *
108  *                                                              *
109  *   Added jump to 'start' so that the program load page        *
110  *   is easily available in the program's third byte.           *
111  *                                                              *
112  *   10/20/04:                                                  *
113  *                                                              *
114  *   Removed call to HOME since screen initialization is        *
115  *   now done in Boot ROM.                                      *
116  *                                                              *
117  *   10/18/04:                                                  *
118  *                                                              *
119  *   Added fix-up code to allow NADANET code to be loaded       *
120  *   anywhere.  Code will get actual load page from $3CF        *
121  *   and patch up to 256 bytes of code by using a unique        *
122  *   fake NADANET page number as a fix-up indicator.            *
123  *                                                              *
124  *   10/16/04:                                                  *
125  *                                                              *
126  *   Fixed ID prefixing.                                        *
127  *                                                              *
128  *   Added 25ms delay in GETMSG and PUTMSG retry loop to        *
129  *   reduce net busy time and allow SERVER to timeout           *
130  *   each iteration (to improve approximate timekeeping).       *
131  *                                                              *
132  *   Changed to new NADAUSER definitions include file.          *
133  *                                                              *
134  *   10/06/04:                                                  *
135  *                                                              *
136  *   Changed chout and flush to prefix machine ID only          *
137  *   when CR caused flush.                                      *
138  *                                                              *
139  *   10/02/04:                                                  *
140  *                                                              *
141  *   Numerous changes and bug fixes.                            *
142  *                                                              *
143  *   09/29/04:                                                  *
144  *                                                              *
145  *   Initial version.                                           *
146  *                                                              *
147  ****************************************************************
```

```
150
151  * Apple ][ definitions
152
153  keybd    equ   $C000      ; Keyboard port
154  VBL      equ   $C019      ; Vertical blanking
155  spkr     equ   $C030      ; Speaker toggle
156  ptrig    equ   $C070      ; Paddle trigger
157
158  dsk6off  equ   $C0E8      ; Deselect 5.25" disk in slot 6
159
160  * Apple II ROM-related addresses
161
162  CH       equ   $24        ; Horizontal cursor position
163  BASL     equ   $28        ; Text line base address
164  CSW      equ   $36        ; Character output vector
165  KSW      equ   $38        ; Keyboard input vector
166  COUT1    equ   $FDF0      ; ROM video out routine
167  MONZ     equ   $FF69      ; Monitor entry point
```

```
                 169  loadpnt  equ   $7800      ; Fake NADANET load point
                 170           put   NADAUSER
                >1    ************************************************************
                >2    *                                                          *
                >3    *         NadaNet Definitions for Applications             *
                >4    *                                                          *
                >5    *            Michael J. Mahon - Oct 14, 2004               *
                >6    *                 Revised Apr 29, 2010                     *
                >7    *                                                          *
                >8    *         Copyright (c) 2004, 2008, 2009, 2010             *
                >9    *                                                          *
                >10   ************************************************************
                >11
                >12   version  equ   $31        ; NadaNet v3.1
                >13
                >14   ********** Control Packet Definition **********
                >15
                >16            dum   0          ; Control packet format:
0000: 00        >17   rqmd     ds    1          ;   Request & Modifier
0001: 00        >18   frmc     ds    1          ;   Complement of sending ID
0002: 00        >19   dst      ds    1          ;   Destination ID (0 = bcast)
0003: 00        >20   frm      ds    1          ;   Sending ID (never 0)
0004: 00 00     >21   adr      ds    2          ;   Address field
0006: 00 00     >22   len      ds    2          ;   Length field
                >23                             ; ========================
                >24   lenctl   ds    0          ; Length of control packet
                >25            dend
                >26
                >27   * Request codes (upper 5 bits) and modifiers (lower 3 bits)
                >28
                >29   reqfac   equ   8          ; Request code factor (2^3)
                >30   reqmask  equ   256-reqfac ; Request code mask (7..3)
                >31   modmask  equ   reqfac-1   ; Modifier code mask (2..0)
                >32
                >33            dum   reqfac     ; Request codes (0 invalid):
0008: 00 00 00  >34   r_PEEK   ds    reqfac     ;   PEEK request
0010: 00 00 00  >35   r_POKE   ds    reqfac     ;   POKE request
0018: 00 00 00  >36   r_CALL   ds    reqfac     ;   CALL request
0020: 00 00 00  >37   r_PUTMSG ds    reqfac     ;   PUTMSG request
0028: 00 00 00  >38   r_GETMSG ds    reqfac     ;   GETMSG request
0030: 00 00 00  >39   r_GETID  ds    reqfac     ;   GETID request
0038: 00 00 00  >40   r_BOOT   ds    reqfac     ;   BOOT request
0040: 00 00 00  >41   r_BCAST  ds    reqfac     ;   BCAST request
0048: 00 00 00  >42   r_BPOKE  ds    reqfac     ;   Broadcast POKE request
0050: 00 00 00  >43   r_PKINC  ds    reqfac     ;   PEEK & INCrement request
0058: 00 00 00  >44   r_PKPOK  ds    reqfac     ;   PEEKPOKE request
0060: 00 00 00  >45   r_RUN    ds    reqfac     ;   RUN request
0068: 00 00 00  >46   r_BRUN   ds    reqfac     ;   BRUN request
                >47                             ; ========================
                >48   maxreq   ds    0          ;   Max request + reqfac
                >49            dend
                >50
```

```
                >51             dum   1           ; Modifier codes (0 invalid):
0001: 00        >52   rm_REQ    ds    1           ;   Request
0002: 00        >53   rm_ACK    ds    1           ;   Acknowledge
0003: 00        >54   rm_DACK   ds    1           ;   Data Acknowledge
0004: 00        >55   rm_NAK    ds    1           ;   Negative Acknowledge
                >56             dend
                >57
                >58   ***************** BCAST tags ******************
                >59   *                                             *
                >60   * High byte of BCAST address field.  Tags <$D0 *
                >61   * can be confused with RAM addresses. (The low *
                >62   * byte may be an additional specification.)    *
                >63   *                                             *
                >64   ***********************************************
                >65
                >66   t_BASIC  equ   $E0         ; Applesoft BASIC program
                >67   t_SYNTH  equ   $F0         ; Crate SYNTH program
                >68   t_VOICE  equ   $F1         ; Crate SYNTH voice
                >69
                >70   ************ NadaNet Page 3 Vector *************
                >71
                >72             dum   $3CC        ; Fixed memory vector
03CC: 00        >73   bootself db    0           ; Machine ID from BOOT
03CD: 4C 00 00  >74   warmstrt jmp   0*0         ; Warm start SERVE loop entry
                >75   nadapage equ   *-1         ; NADANET load page
                >76             dend
                >77
                >78   ***************** Entry points *****************
                >79
                >80             dum   loadpnt     ; NadaNet load address
                >81
7800: 20 00 78  >82   entry    jsr   *            ; BOOT entry: init and
7803: 20 03 78  >83   servelp  jsr   *            ;  Run request server
7806: 4C 03 78  >84            jmp   servelp      ;    forever...
7809: 4C 09 78  >85   init     jmp   *            ; Initialize and return
780C: 4C 0C 78  >86   serve    jmp   *            ; Run reqest server
780F: 4C 0F 78  >87   peek     jmp   *            ; Peek/Poke 'sbuf+dst' for
7812: 4C 12 78  >88   poke     jmp   *            ;  'sbuf+len' bytes at 'sbuf+adr'
7815: 4C 15 78  >89   call     jmp   *            ; Call 'sbuf+dst' at 'sbuf+adr'
7818: 4C 18 78  >90   putmsg   jmp   *            ; Put message to server
781B: 4C 1B 78  >91   getmsg   jmp   *            ; Get message from server
781E: 4C 1E 78  >92   bcast    jmp   *            ; Broadcast data
7821: 4C 21 78  >93   bpoke    jmp   *            ; Broadcast 2-byte POKE
7824: 4C 24 78  >94   peekinc  jmp   *            ; PEEK & INC 2-byte val
7827: 4C 27 78  >95   peekpoke jmp   *            ; PEEKPOKE 2-byte val
782A: 4C 2A 78  >96   run      jmp   *            ; RUN Applseoft prog
782D: 4C 2D 78  >97   brun     jmp   *            ; BRUN M/L prog
7830: 4C 30 78  >98   rcvctl   jmp   *            ; Receive control pkt
7833: 4C 33 78  >99   rcvptr   jmp   *            ; Receive to 'ptr'
7836: 4C 36 78 >100   rarl=>al jmp   *            ; Rbuf adr,len=>address,length
7839: 4C 39 78 >101   rcvlong  jmp   *            ; Receive long data
```

```
                    >103  ********** Parameters and variables ************
                    >104
783C: 00            >105  self     db    0          ; Our own machine ID
783D: 00 00 00      >106  sbuf     ds    lenctl     ; Control pkt send buffer
7845: 00 00 00      >107  rbuf     ds    lenctl     ; Control pkt receive buffer
784D: 00 00         >108  locaddr  dw    0          ; Local address of req data
784F: 00            >109  retrylim db    0          ; Limit of REQUEST resends
7850: 00            >110  servecnt db    0          ; SERVE iterations (0=256)
                    >111
                    >112  parmsiz  equ   *-self     ; Size of parameter area
                    >113
                    >114  ************ Counters and Version **************
                    >115
7851: 00            >116  arbxv    db    0          ; Arbitrate X iters (modified)
7852: 00            >117  tolim    db    0          ; RCVPKT timeout limit
7853: 08            >118  reqctr   db    8          ; SERVER request counter
7854: 00            >119  reqretry db    0          ; xxxREQ retries remaining
7855: 00            >120  retrycnt db    0          ; REQUEST resend count
7856: 00 00         >121  errprot  dw    0          ; Protocol error count
7858: 00 00         >122  ckerr    dw    0          ; Checksum error count
785A: 00 00         >123  frmcerr  dw    0          ; 'frmc' collision errors
785C: 31            >124  nadaver  db    version    ; NadaNet version
                    >125
                    >126  * Table of allocated machine IDs (allocated = non-zero)
                    >127  *        (Only present in "master" machines)
                    >128
                    >129  maxid    equ   31         ; Maximum number of machines
                    >130
785D: 1F            >131  idtable  db    maxid      ; Table of machine attributes
785E: 00 00 00      >132           ds    maxid      ; Rest of ID table (=0)
                    >133
                    >134           dend
```

```
 172           use   NADAMACS
>1    ****************** Macro definitions ********************
>2
>3    inc16    mac
>4             inc   ]1          ; Increment 16-bit word.
>5             do    ]1+1/$100   ; If ]1 is non-page zero
>6             bne   *+5         ; - No carry.
>7             else              ; Else if ]1 on page zero
>8             bne   *+4         ; - No carry.
>9             fin
>10            inc   ]1+1        ; Propagate carry.
>11            eom
>12
>13   mov16    mac
>14            lda   ]1          ; Move 2 bytes
>15            sta   ]2
>16            if    #=]1
>17            lda   ]1/$100     ; high byte of immediate
>18            else
>19            lda   1+]1
>20            fin
>21            sta   1+]2
>22            eom
>23
>24   delay    mac
>25            ldx   #]1/5       ; (5 cycles per iteration)
>26   ]delay   dex
>27            bne   ]delay
>28            eom
>29
>30   dlyms    mac
>31            ldy   #]1         ; Delay 1ms. per iteration
>32   ]dly     delay 1020-4      ; Cycles per ms. - 4
>33            dey
>34            bne   ]dly
>35            eom
>36
>37   align    mac
>38            ds    *-1/]1*]1+]1-*
>39            eom
>40
```

```
                       173  ***** Circular Buffer Control Block Definition *****
                       174
                       175           dum   0           ; Control Block layout:
0000: 00 00            176  lok      ds    2           ;   Lock word (0=unlocked)
0002: 00               177  tlx      ds    1           ;   Tail index
0003: 00               178  hdx      ds    1           ;   Head index
0004: 00 00            179  buf      ds    2           ;   Ptr to 256-byte buffer
                       180                            ; ======================
                       181  CBCBlen  ds    0           ; Length of CBCB
                       182           dend
                       183
                       184           org   $6000
6000: 4C 27 60         185  enter    jmp   relay       ; Standard entry on load page
6003: 4C DF 61         186  DETACH   jmp   detach      ; DETACH entry point
                       187
                       188  * Parameter area
                       189
6006: 00 00            190  oucbadr  da    0*0         ; Ptr to output CBCB in master
6008: 00 00 00         191  incb     ds    CBCBlen-2   ; Local input CB control block
600C: 7D 62            192           da    inbuf       ; ptr to input CB data
                       193
                       194  * Definitions
                       195
                       196  buflen   equ   $76         ; Length of outbuf
                       197
                       198  address  equ   $FC         ; Page 0 scratch pointer
                       199
                       200  * DOS and ProDOS I/O vector addresses
                       201
                       202  DOShooks equ   $AA53
                       203  Prohooks equ   $BE30
                       204
                       205  * Non-Page Zero variables
                       206
600E: 00 00 00         207  mcb      ds    CBCBlen     ; Local copy of master CBCB
                       208  ctlid    equ   enter+2     ; Controlling machine's ID
6014: 00               209  savex    db    0           ; Register save temp
6015: 00               210  savey    db    0           ; Register save temp
6016: 00               211  outlen   db    0           ; Points to next outbuf char
6017: 00               212  inch     db    0           ; Previous input character
6018: 00 00            213  hookadr  da    0           ; Address of OS I/O hooks
                       214
601A: 00 00            215  origCSW  dw    0           ; CSW at ATTACH
601C: 00 00            216           dw    0           ; KSW at ATTACH
                       217
601E: 4C F0 FD         218  prevCOUT jmp   COUT1       ; I/O hooks are saved
6021: CD 03            219           da    warmstrt    ;  here if an OS is active.
                       220
6023: 2A 60            221  hooks    da    chout       ; ATTACH hook vector
6025: AD 61            222           da    keyin
                       223
6027: 4C 07 62         224  relay    jmp   start       ; (entry vector on load page)
```

```
                    226  ************************************************************
                    227  *                                                          *
                    228  *                        C H O U T                         *
                    229  *                                                          *
                    230  *              Michael J. Mahon - Sep 29, 2004             *
                    231  *                    Revised Apr 30, 2010                  *
                    232  *                                                          *
                    233  *           Copyright (c) 2004, 2007, 2009, 2010           *
                    234  *                                                          *
                    235  *   CHOUT adds output characters to outbuf, suppressing    *
                    236  *   local input echo, and flushes the buffer to the        *
                    237  *   controlling machine whenever a CR is received or the   *
                    238  *   buffer is filled.                                      *
                    239  *                                                          *
                    240  ************************************************************
                    241
602A: 48            242  chout    pha              ; Save output char.
602B: 20 1E 60      243           jsr   prevCOUT   ; Send to local video display
602E: CD 17 60      244           cmp   inch       ; Is this an input echo?
6031: D0 07         245           bne   :out       ; -No, process it.
6033: A9 00         246           lda   #0         ; -Yes, clear the
6035: 8D 17 60      247           sta   inch       ;   input character
6038: 68            248           pla              ;    restore A
6039: 60            249           rts              ;     and return.
                    250
603A: 8C 15 60      251  :out     sty   savey      ; Save Y
603D: 8E 14 60      252           stx   savex      ;  and X.
6040: A0 00         253           ldy   #0         ; Clear input
6042: 8C 17 60      254           sty   inch       ;  character.
6045: AC 16 60      255           ldy   outlen     ; Add char
6048: 99 07 62      256           sta   outbuf,y   ;  to output buffer
604B: C8            257           iny              ;   and bump pointer.
604C: C9 8D         258           cmp   #$8D       ; Carriage Return?
604E: F0 04         259           beq   :cr        ; -Yes, flush.
6050: C0 76         260           cpy   #buflen    ; -No. outbuf full?
6052: 90 03         261           bcc   :exit      ; -No, exit.
6054: 20 62 60      262  :cr      jsr   flush      ; -Yes, flush it.
6057: 8C 16 60      263  :exit    sty   outlen     ; Update outlen
605A: AC 15 60      264           ldy   savey      ; Restore Y
605D: AE 14 60      265           ldx   savex      ;  and X
6060: 68            266           pla              ;   and A
6061: 60            267           rts              ; Return.
                    268
                    269
6062: 8C 16 60      270  flush    sty   outlen     ; Save output length
6065: A0 01         271           ldy   #1         ; Set timeout to 1
6067: 8C 4F 78      272           sty   retrylim   ;  cycle of 3 retries.
                    273  :puttoCB mov16 oucbadr;sbuf+adr ; Lock output CB
606A: AD 06 60      273           lda   oucbadr    ; Move 2 bytes
606D: 8D 41 78      273           sta   sbuf+adr
6070: AD 07 60      273           lda   1+oucbadr
6073: 8D 42 78      273           sta   1+sbuf+adr
```

```
                    273           eom
                    274           mov16 #1;sbuf+len ; in controlling machine.
6076: A9 01         274           lda   #1          ; Move 2 bytes
6078: 8D 43 78      274           sta   sbuf+len
607B: A9 00         274           lda   #1/$100     ; high byte of immediate
607D: 8D 44 78      274           sta   1+sbuf+len
                    274           eom
6080: AD 02 60      275           lda   ctlid
6083: 8D 3F 78      276           sta   sbuf+dst
6086: 20 27 78      277           jsr   peekpoke
6089: B0 08         278           bcs   :serve      ; Serve on failure.
608B: AD 4B 78      279           lda   rbuf+len    ; Did we get
608E: 0D 4C 78      280           ora   rbuf+len+1  ;  the lock?
6091: F0 0B         281           beq   :locked     ; -Yes!
6093: A9 02         282  :serve   lda   #2          ; -No, serve a request.
6095: 8D 50 78      283           sta   servecnt
6098: 20 0C 78      284           jsr   serve
609B: 4C 6A 60      285           jmp   :puttoCB    ;   and try again.
                    286
                    287  :locked  mov16 #CBCBlen-tlx;sbuf+len ; Get CB ctl block
609E: A9 04         287           lda   #CBCBlen-tlx ; Move 2 bytes
60A0: 8D 43 78      287           sta   sbuf+len
60A3: A9 00         287           lda   #CBCBlen-tlx/$100 ; high byte of immediate
60A5: 8D 44 78      287           sta   1+sbuf+len
                    287           eom
60A8: 18            288           clc
60A9: AD 06 60      289           lda   oucbadr     ; sbuf+adr = oucbadr+tlx
60AC: 69 02         290           adc   #tlx
60AE: 8D 41 78      291           sta   sbuf+adr
60B1: AD 07 60      292           lda   oucbadr+1
60B4: 69 00         293           adc   #0
60B6: 8D 42 78      294           sta   sbuf+adr+1
                    295           mov16 #mcb+tlx;locaddr
60B9: A9 10         295           lda   #mcb+tlx    ; Move 2 bytes
60BB: 8D 4D 78      295           sta   locaddr
60BE: A9 60         295           lda   #mcb+tlx/$100 ; high byte of immediate
60C0: 8D 4E 78      295           sta   1+locaddr
                    295           eom
60C3: 20 0F 78      296           jsr   peek
60C6: B0 0C         297           bcs   :unlock     ; Unlock & retry on failure.
60C8: 18            298           clc               ; Compute space avail
60C9: AD 11 60      299           lda   mcb+hdx     ;  = hdx - tlx - 1
60CC: ED 10 60      300           sbc   mcb+tlx
60CF: CD 16 60      301           cmp   outlen      ; Will data fit?
60D2: B0 36         302           bcs   :fits       ; -Yes.
                    303  :unlock  mov16 #2;sbuf+len ;-No, release lock.
60D4: A9 02         303           lda   #2          ; Move 2 bytes
60D6: 8D 43 78      303           sta   sbuf+len
60D9: A9 00         303           lda   #2/$100     ; high byte of immediate
60DB: 8D 44 78      303           sta   1+sbuf+len
                    303           eom
                    304           mov16 oucbadr;sbuf+adr
```

```
60DE: AD 06 60  304           lda   oucbadr    ; Move 2 bytes
60E1: 8D 41 78  304           sta   sbuf+adr
60E4: AD 07 60  304           lda   1+oucbadr
60E7: 8D 42 78  304           sta   1+sbuf+adr
                304           eom
                305           mov16 #mcb;locaddr
60EA: A9 0E     305           lda   #mcb       ; Move 2 bytes
60EC: 8D 4D 78  305           sta   locaddr
60EF: A9 60     305           lda   #mcb/$100  ; high byte of immediate
60F1: 8D 4E 78  305           sta   1+locaddr
                305           eom
60F4: AD 02 60  306           lda   ctlid
60F7: 8D 3F 78  307           sta   sbuf+dst
60FA: 20 12 78  308           jsr   poke       ; Release MCB lock,
60FD: 90 94     309           bcc   :serve     ;  serve, and try again.
60FF: A9 02     310           lda   #2         ; On failure, serve
6101: 8D 50 78  311           sta   servecnt   ;  for 40 ms. and
6104: 20 0C 78  312           jsr   serve
6107: 4C D4 60  313           jmp   :unlock    ;   try to unlock again.
                314
610A: 18        315  :fits    clc              ; Set sbuf+adr
610B: AD 12 60  316           lda   mcb+buf    ;  = (mcb+buf) + tlx
610E: 6D 10 60  317           adc   mcb+tlx
6111: 8D 41 78  318           sta   sbuf+adr
6114: AD 13 60  319           lda   mcb+buf+1
6117: 69 00     320           adc   #0
6119: 8D 42 78  321           sta   sbuf+adr+1
                322           mov16 #outbuf;locaddr
611C: A9 07     322           lda   #outbuf    ; Move 2 bytes
611E: 8D 4D 78  322           sta   locaddr
6121: A9 62     322           lda   #outbuf/$100 ; high byte of immediate
6123: 8D 4E 78  322           sta   1+locaddr
                322           eom
6126: 38        323           sec              ; Compute length to end
6127: A9 00     324           lda   #0         ;  of MCB = 256 - tlx
6129: ED 10 60  325           sbc   mcb+tlx
612C: CD 16 60  326           cmp   outlen     ; Does string wrap buffer?
612F: 90 03     327           bcc   :wrap      ; -Yes, save 1st chunk length
6131: AD 16 60  328           lda   outlen     ; -No, send in one chunk.
6134: 8D 43 78  329  :wrap    sta   sbuf+len   ; Length of 1st chunk (L1)
6137: 20 12 78  330           jsr   poke       ; POKE first part
613A: 18        331           clc
613B: AD 4D 78  332           lda   locaddr
613E: 6D 43 78  333           adc   sbuf+len   ; Advance locaddr by L1
6141: 8D 4D 78  334           sta   locaddr
6144: AD 4E 78  335           lda   locaddr+1
6147: 69 00     336           adc   #0
6149: 8D 4E 78  337           sta   locaddr+1
614C: 38        338           sec
614D: AD 16 60  339           lda   outlen
6150: ED 43 78  340           sbc   sbuf+len   ; outlen > L1?
6153: F0 12     341           beq   :done      ; -No, we're done.
```

```
6155: 8D 43 78  342           sta   sbuf+len   ; -Yes, set 2nd chunk length
                343           mov16 mcb+buf;sbuf+adr
6158: AD 12 60  343           lda   mcb+buf    ; Move 2 bytes
615B: 8D 41 78  343           sta   sbuf+adr
615E: AD 13 60  343           lda   1+mcb+buf
6161: 8D 42 78  343           sta   1+sbuf+adr
                343           eom
6164: 20 12 78  344           jsr   poke       ; POKE wrapped chunk.
6167: 18        345  :done    clc
6168: AD 16 60  346           lda   outlen     ; Update tlx
616B: 6D 10 60  347           adc   mcb+tlx    ;  = tlx + total length
616E: 8D 10 60  348           sta   mcb+tlx
                349  :rerel   mov16 #tlx+1;sbuf+len ; Length of (lock + tlx)
6171: A9 03     349           lda   #tlx+1     ; Move 2 bytes
6173: 8D 43 78  349           sta   sbuf+len
6176: A9 00     349           lda   #tlx+1/$100 ; high byte of immediate
6178: 8D 44 78  349           sta   1+sbuf+len
                349           eom
                350           mov16 oucbadr;sbuf+adr
617B: AD 06 60  350           lda   oucbadr    ; Move 2 bytes
617E: 8D 41 78  350           sta   sbuf+adr
6181: AD 07 60  350           lda   1+oucbadr
6184: 8D 42 78  350           sta   1+sbuf+adr
                350           eom
                351           mov16 #mcb;locaddr
6187: A9 0E     351           lda   #mcb       ; Move 2 bytes
6189: 8D 4D 78  351           sta   locaddr
618C: A9 60     351           lda   #mcb/$100  ; high byte of immediate
618E: 8D 4E 78  351           sta   1+locaddr
                351           eom
6191: AD 02 60  352           lda   ctlid
6194: 8D 3F 78  353           sta   sbuf+dst
6197: 20 12 78  354           jsr   poke       ; Release lock & update tlx.
619A: 90 0B     355           bcc   :exit      ; -OK
619C: A9 02     356           lda   #2         ; -NG, serve
619E: 8D 50 78  357           sta   servecnt   ;   for 40 ms.
61A1: 20 0C 78  358           jsr   serve      ;    and
61A4: 4C 71 61  359           jmp   :rerel     ;     try release again.
                360
61A7: A0 00     361  :exit    ldy   #0         ; Set outbuf empty.
61A9: 8C 16 60  362           sty   outlen
61AC: 60        363           rts
```

```
                        ********************************************************
                        *                                                      *
                        *                      K E Y I N                       *
                        *                                                      *
                        *            Michael J. Mahon - Sep 29, 2004           *
                        *                  Revised Dec 10, 2008                *
                        *                                                      *
                        *             Copyright (c) 2004, 2007, 2008           *
                        *                                                      *
                        *   KEYIN first checks the output buffer.  If it is not  *
                        *   empty, it flushes it (and any prompt).  It then looks*
                        *   in the input buffer for an imput character.  If the  *
                        *   input circular buffer is empty, it serves until more *
                        *   input is received.                                 *
                        *                                                      *
                        ********************************************************
                        
61AD: A4 24             keyin    ldy   CH          ; Horizontal cursor
61AF: 91 28                      sta   (BASL),y    ; Restore char at cursor postion
61B1: 8E 14 60                   stx   savex       ; Save X
61B4: AC 16 60                   ldy   outlen      ; Is outbuf empty?
61B7: F0 03                      beq   :getchr     ; -Yes, do input.
61B9: 20 62 60                   jsr   flush       ; -No, flush it.
61BC: AD 08 60          :getchr  lda   incb+lok    ; Is input CB
61BF: 0D 09 60                   ora   incb+lok+1  ;  locked?
61C2: F0 06                      beq   :ok         ; -No, check for input.
61C4: 20 0C 78          :wait    jsr   serve       ; -Yes, serve a request
61C7: 4C BC 61                   jmp   :getchr     ;  and try again.
                        
61CA: AC 0B 60          :ok      ldy   incb+hdx    ; Index to head of CB
61CD: CC 0A 60                   cpy   incb+tlx    ; CB empty?
61D0: F0 F2                      beq   :wait       ; -Yes, wait for input.
61D2: EE 0B 60                   inc   incb+hdx    ; -No, inc head index
61D5: B9 7D 62                   lda   inbuf,y     ;   and get next character.
61D8: 8D 17 60                   sta   inch        ; Save for echo suppression
61DB: AE 14 60                   ldx   savex       ; Restore X
61DE: 60                         rts               ;  and return with char in A.
```

```
                        403  ************************************************************
                        404  *                                                          *
                        405  *                      D E T A C H                         *
                        406  *                                                          *
                        407  *            Michael J. Mahon - Sep 29, 2004               *
                        408  *                  Revised Jan 29, 2009                    *
                        409  *                                                          *
                        410  *          Copyright (c) 2004, 2007, 2008, 2009            *
                        411  *                                                          *
                        412  *   Detach from this machine by restoring the OS hooks     *
                        413  *   (if any) and the original CSW/KSW hooks, then re-      *
                        414  *   enter serverlp, restoring the original state of the    *
                        415  *   machine.                                               *
                        416  *                                                          *
                        417  ************************************************************
                        418
                        419  detach   mov16 hookadr;address ; Point at hooks
61DF: AD 18 60          419            lda   hookadr    ; Move 2 bytes
61E2: 85 FC             419            sta   address
61E4: AD 19 60          419            lda   1+hookadr
61E7: 85 FD             419            sta   1+address
                        419            eom
61E9: A0 03             420            ldy   #3
61EB: B9 1F 60          421  :restore lda   prevCOUT+1,y ; Restore previous
61EE: 91 FC             422            sta   (address),y  ;  I/O hooks and
61F0: B9 1A 60          423            lda   origCSW,y  ;     original CSW/KSW
61F3: 99 36 00          424            sta   CSW,y      ;      vectors.
61F6: 88                425            dey
61F7: 10 F2             426            bpl   :restore
61F9: 8C 00 60          427            sty   enter      ; Clobber "attached" signature.
61FC: 8C 50 78          428            sty   servecnt   ; Set servecnt back to max.
61FF: A0 32             429            ldy   #50        ; Set "timeout" back
6201: 8C 4F 78          430            sty   retrylim   ;  to default.
6204: 4C CD 03          431            jmp   warmstrt   ; and re-enter 'servelp'.
```

```
                433  * Output and input buffers
                434
                435  outbuf   equ   *          ; Output buffer
                436  inbuf    equ   *+buflen   ; 256-byte circular buffer
                437
                438  ************************************************************
                439  *                                                          *
                440  *   Code to fix up references to NadaNet param area and    *
                441  *   ATTACH hooks for remote operation of the machine.      *
                442  *                                                          *
                443  *   (Following code is overwritten by outbuf and inbuf.)   *
                444  *                                                          *
                445  ************************************************************
                446
                447  ]iter    equ   outbuf-flush/2 ; # of iters needed
                448           err   ]iter/256  ; Reloc limited to 2 pages.
                449
6207: A0 D2     450  start    ldy   #]iter     ; Fix up references to
6209: AE CF 03  451           ldx   nadapage   ;  fake NADANET load page
620C: 8E 7B 62  452           stx   reloc+2    ; Special ATTACH fix-up
620F: B9 62 60  453  :loop    lda   flush,y    ; Get byte in first half
6212: C9 78     454           cmp   #>loadpnt  ; Fake page?
6214: D0 04     455           bne   :sk1       ; -No, skip this one.
6216: 8A        456           txa              ; -Yes, patch it with
6217: 99 62 60  457           sta   flush,y    ;   the real page #.
621A: B9 34 61  458  :sk1     lda   flush+]iter,y ; (same for second half)
621D: C9 78     459           cmp   #>loadpnt  ; Fake page?
621F: D0 04     460           bne   :skip      ; -No, skip this one.
6221: 8A        461           txa              ; -Yes, patch it with
6222: 99 34 61  462           sta   flush+]iter,y ; the real page #.
6225: 88        463  :skip    dey
6226: D0 E7     464           bne   :loop      ; (fall into ATTACH)
```

```
                466  ************************************************************
                467  *                                                          *
                468  *                      A T T A C H                         *
                469  *                                                          *
                470  *            Michael J. Mahon - Sep 29, 2004               *
                471  *                  Revised Dec 19, 2008                    *
                472  *                                                          *
                473  *                Copyright (c) 2004, 2008                  *
                474  *                                                          *
                475  *   ATTACH saves the CSW/KSW vectors for restoration by     *
                476  *   'detach', then saves the character output routine if    *
                477  *   a disk OS is active and 'chout' is not _already_        *
                478  *   installed, then sets the output and input hooks so      *
                479  *   that the OS is still connected.                        *
                480  *                                                          *
                481  *   It then enters the ROM Monitor under remote control.   *
                482  *                                                          *
                483  ************************************************************
                484
                485  ATTACH   mov16 #CSW;address ; Default hook address
6228: A9 36     485           lda   #CSW       ; Move 2 bytes
622A: 85 FC     485           sta   address
622C: A9 00     485           lda   #CSW/$100  ; high byte of immediate
622E: 85 FD     485           sta   1+address
                485           eom
6230: AE CF 03  486           ldx   nadapage   ; What environment?
6233: E0 91     487           cpx   #$91       ; ProDOS machine?
6235: D0 0A     488           bne   :ckdos     ; -No, check if DOS.
                489           mov16 #Prohooks;address ; -Yes, ProDOS hooks.
6237: A9 30     489           lda   #Prohooks  ; Move 2 bytes
6239: 85 FC     489           sta   address
623B: A9 BE     489           lda   #Prohooks/$100 ; high byte of immediate
623D: 85 FD     489           sta   1+address
                489           eom
623F: D0 0C     490           bne   :ckhook    ; (always)
                491
6241: E0 8D     492  :ckdos   cpx   #$8D       ; DOS machine?
6243: D0 08     493           bne   :ckhook    ; -No, use default hooks.
                494           mov16 #DOShooks;address ; -Yes, DOS hooks.
6245: A9 53     494           lda   #DOShooks  ; Move 2 bytes
6247: 85 FC     494           sta   address
6249: A9 AA     494           lda   #DOShooks/$100 ; high byte of immediate
624B: 85 FD     494           sta   1+address
                494           eom
624D: A0 01     495  :ckhook  ldy   #1
624F: B1 FC     496           lda   (address),y ; If active 'CSW'
6251: C9 60     497           cmp   #>chout    ;   is already us,
6253: F0 1A     498           beq   :sethook   ;    then just set hookadr.
6255: A0 03     499           ldy   #3         ; Else save it as the
6257: B1 FC     500  :save    lda   (address),y ; 'local echo' routine
6259: 99 1F 60  501           sta   prevCOUT+1,y ;  along with 'KSW' addr.
625C: 88        502           dey
```

```
625D: 10 F8     503           bpl   :save
625F: A0 03     504           ldy   #3          ; Set hooks for ATTACH, and
6261: B9 36 00  505  :saveset lda   CSW,y       ;  save original CSW/KSW
6264: 99 1A 60  506           sta   origCSW,y   ;   vectors for detach.
6267: B9 23 60  507           lda   hooks,y
626A: 91 FC     508           sta   (address),y
626C: 88        509           dey
626D: 10 F2     510           bpl   :saveset
                511  :sethook mov16 address;hookadr ; Save addr of hooks
626F: A5 FC     511           lda   address    ; Move 2 bytes
6271: 8D 18 60  511           sta   hookadr
6274: A5 FD     511           lda   1+address
6276: 8D 19 60  511           sta   1+hookadr
                511           eom
6279: AD 48 78  512  reloc    lda   rbuf+frm   ; Get controlling machine's
627C: 8D 02 60  513           sta   ctlid      ;  ID and save it.
627F: 4C 69 FF  514           jmp   MONZ       ; Enter the Monitor.


--End assembly, 642 bytes, Errors: 0


Symbol table - alphabetical order:

?  ATTACH  =$6228      BASL    =$28        CBCBlen =$06        CH      =$24
   COUT1   =$FDF0      CSW     =$36     ?  DETACH  =$6003      DOShooks=$AA53
?  KSW     =$38        MONZ    =$FF69      Prohooks=$BE30   ?  VBL     =$C019
V  ]iter   =$D2        address =$FC        adr     =$04     MD?align   =$8000
?  arbxv   =$7851   ?  bcast   =$781E   ?  bootself=$03CC   ?  bpoke   =$7821
?  brun    =$782D      buf     =$04        buflen  =$76     ?  call    =$7815
   chout   =$602A   ?  ckerr   =$7858      ctlid   =$6002   MD?delay   =$8000
   detach  =$61DF   MD?dlyms   =$8000   ?  dsk6off =$C0E8      dst     =$02
   enter   =$6000   ?  entry   =$7800   ?  errprot =$7856      flush   =$6062
   frm     =$03     ?  frmc    =$01     ?  frmcerr =$785A   ?  getmsg  =$781B
   hdx     =$03        hookadr =$6018      hooks   =$6023   ?  idtable =$785D
   inbuf   =$627D   MD?inc16   =$8000      incb    =$6008      inch    =$6017
?  init    =$7809   ?  keybd   =$C000      keyin   =$61AD      len     =$06
   lenctl  =$08        loadpnt =$7800      locaddr =$784D      lok     =$00
   maxid   =$1F     ?  maxreq  =$70        mcb     =$600E   ?  modmask =$07
MD mov16   =$8000      nadapage=$03CF   ?  nadaver =$785C      origCSW =$601A
   oucbadr =$6006      outbuf  =$6207      outlen  =$6016   ?  parmsiz =$15
   peek    =$780F   ?  peekinc =$7824      peekpoke=$7827      poke    =$7812
   prevCOUT=$601E   ?  ptrig   =$C070   ?  putmsg  =$7818   ?  r_BCAST =$40
?  r_BOOT  =$38     ?  r_BPOKE =$48     ?  r_BRUN  =$68     ?  r_CALL  =$18
?  r_GETID =$30     ?  r_GETMSG=$28     ?  r_PEEK  =$08     ?  r_PKINC =$50
?  r_PKPOK =$58     ?  r_POKE  =$10     ?  r_PUTMSG=$20     ?  r_RUN   =$60
?  rarl=>al=$7836      rbuf    =$7845   ?  rcvctl  =$7830   ?  rcvlong =$7839
?  rcvptr  =$7833      relay   =$6027      reloc   =$6279   ?  reqctr  =$7853
   reqfac  =$08     ?  reqmask =$F8     ?  reqretry=$7854   ?  retrycnt=$7855
   retrylim=$784F   ?  rm_ACK  =$02     ?  rm_DACK =$03     ?  rm_NAK  =$04
?  rm_REQ  =$01     ?  rqmd    =$00     ?  run     =$782A      savex   =$6014
   savey   =$6015      sbuf    =$783D      self    =$783C      serve   =$780C
```

```
   servecnt=$7850      servelp =$7803   ?  spkr    =$C030      start   =$6207
?  t_BASIC =$E0     ?  t_SYNTH =$F0     ?  t_VOICE =$F1        tlx     =$02
?  tolim   =$7852      version =$31        warmstrt=$03CD
```

Symbol table - numerical order:

```
?  rqmd    =$00        lok     =$00     ?  frmc    =$01     ?  rm_REQ  =$01
   dst     =$02     ?  rm_ACK  =$02        tlx     =$02        frm     =$03
?  rm_DACK =$03        hdx     =$03        adr     =$04     ?  rm_NAK  =$04
   buf     =$04        len     =$06        CBCBlen =$06     ?  modmask =$07
   lenctl  =$08        reqfac  =$08     ?  r_PEEK  =$08     ?  r_POKE  =$10
?  parmsiz =$15     ?  r_CALL  =$18        maxid   =$1F     ?  r_PUTMSG=$20
   CH      =$24        BASL    =$28     ?  r_GETMSG=$28     ?  r_GETID =$30
   version =$31        CSW     =$36     ?  KSW     =$38     ?  r_BOOT  =$38
?  r_BCAST =$40     ?  r_BPOKE =$48     ?  r_PKINC =$50     ?  r_PKPOK =$58
?  r_RUN   =$60     ?  r_BRUN  =$68     ?  maxreq  =$70        buflen  =$76
V  ]iter   =$D2     ?  t_BASIC =$E0     ?  t_SYNTH =$F0     ?  t_VOICE =$F1
?  reqmask =$F8        address =$FC     ?  bootself=$03CC      warmstrt=$03CD
   nadapage=$03CF      enter   =$6000      ctlid   =$6002   ?  DETACH  =$6003
   oucbadr =$6006      incb    =$6008      mcb     =$600E      savex   =$6014
   savey   =$6015      outlen  =$6016      inch    =$6017      hookadr =$6018
   origCSW =$601A      prevCOUT=$601E      hooks   =$6023      relay   =$6027
   chout   =$602A      flush   =$6062      keyin   =$61AD      detach  =$61DF
   outbuf  =$6207      start   =$6207   ?  ATTACH  =$6228      reloc   =$6279
   inbuf   =$627D      loadpnt =$7800   ?  entry   =$7800      servelp =$7803
?  init    =$7809      serve   =$780C      peek    =$780F      poke    =$7812
?  call    =$7815   ?  putmsg  =$7818   ?  getmsg  =$781B   ?  bcast   =$781E
?  bpoke   =$7821   ?  peekinc =$7824      peekpoke=$7827   ?  run     =$782A
?  brun    =$782D   ?  rcvctl  =$7830   ?  rcvptr  =$7833   ?  rarl=>al=$7836
?  rcvlong =$7839      self    =$783C      sbuf    =$783D      rbuf    =$7845
   locaddr =$784D      retrylim=$784F      servecnt=$7850   ?  arbxv   =$7851
?  tolim   =$7852   ?  reqctr  =$7853   ?  reqretry=$7854   ?  retrycnt=$7855
?  errprot =$7856   ?  ckerr   =$7858   ?  frmcerr =$785A   ?  nadaver =$785C
?  idtable =$785D   MD?align   =$8000   MD?dlyms   =$8000   MD?delay   =$8000
MD mov16   =$8000   MD?inc16   =$8000      DOShooks=$AA53      Prohooks=$BE30
?  keybd   =$C000   ?  VBL     =$C019   ?  spkr    =$C030   ?  ptrig   =$C070
?  dsk6off =$C0E8      COUT1   =$FDF0      MONZ    =$FF69
```