```
1   ************************************************************
2   *                                                          *
3   *                S Y N T H . L O A D E R                   *
4   *                                                          *
5   *            Michael J. Mahon - Sep 30, 2008               *
6   *                                                          *
7   *       Copyright (c) 1996, 2003, 2004, 2005, 2008         *
8   *                                                          *
9   *   SYNTH.LOADER uses BCAST to load CRATE.SYNTH and its    *
10  *   voices, allowing each voice to be sent only once.      *
11  *                                                          *
12  *   It is &BRUN on each oscillator with the oscillator's   *
13  *   music pre-loaded and its voice list inserted just      *
14  *   after SYNTH.LOADER's entry point.                      *
15  *                                                          *
16  *  SYNTH.LOADER does the following steps:                  *
17  *     1. Clear screen and sign on                          *
18  *     2. Receive BCAST SYNTH code                          *
19  *     3. Display the voice list on screen                  *
20  *     4. Receive BCAST voices, fixing up 'voicetbl',       *
21  *        allocating memory, and marking them loaded        *
22  *     5. When all voices loaded, jump to SYNTH entry       *
23  *                                                          *
24  ************************************************************
25  *                                                          *
26  *                     Change History                       *
27  *                                                          *
28  *   10/06/08:                                              *
29  *                                                          *
30  *   First version, adapted from ROM boot code.             *
31  *                                                          *
32  ************************************************************
```

```
 34   ******************** Version setup ************************
 35
 36            org   $200       ; Load to page 2
 37   master   equ   0          ; Non-master version
 38   dos      equ   0          ; Non-DOS version
 39   crate    equ   1          ; Crate version compatibiluty
 40   mserve   equ   0          ; Non-Message Server version
 41   ROMboot  equ   0          ; ROM boot version
 42   enhboot  equ   0          ; Enhanced //e version
 43
 44           put   NADACONST
>1    * NadaNet Constant definitions
>2
>3    * Apple ][ definitions
>4
>5    keybd    equ   $C000      ; Keyboard port
>6    kbstrobe equ   $C010      ; Keyboard strobe
>7    VBL      equ   $C019      ; Vertical blanking
>8    spkr     equ   $C030      ; Speaker toggle
>9    an0      equ   $C058      ; Annunciator 0 base addr
>10   an1      equ   an0+2
>11   an2      equ   an0+4
>12   an3      equ   an0+6
>13   pb0      equ   $C061      ; "Pushbutton" 0 base addr
>14   pb1      equ   pb0+1
>15   pb2      equ   pb0+2
>16   ptrig    equ   $C070      ; Paddle trigger
>17   dsk6off  equ   $C0E8      ; Deselect 5.25" disk in slot 6
>18
>19   * Apple Monitor definitions
>20
>21   CSW      equ   $36        ; Output vector
>22   KSW      equ   $38        ; Input vector
>23   SOFTEV   equ   $3F2       ; Soft re-entry vector
>24   PWREDUP  equ   $3F4       ; Powered-Up check byte
>25
>26   PRBL2    equ   $F94A      ; Display (X) blanks
>27   PREAD    equ   $FB1E      ; Read PDL(X) into Y
>28   HOME     equ   $FC58      ; Clear display
>29   CROUT1   equ   $FD8B      ; Clear to EOL, then CR
>30   PRBYTE   equ   $FDDA      ; Display A as hex byte
>31   COUT     equ   $FDED      ; Display character in A
>32   BELL     equ   $FF3A      ; Beep for 100ms.
>33
>34   * Applesoft definitions
>35
>36   PSTART   equ   $67        ; Start of BASIC prog
>37   VARTAB   equ   $69        ; End prog / start vars
>38   FRETOP   equ   $6F        ; Start of string storage
>39   HIMEM    equ   $73        ; Highest BASIC mem
>40   PROGEND  equ   $AF        ; End of BASIC prog
>41   ONERR    equ   $D8        ; ONERR flag (0 = off)
```

```
>42
>43   COLDSTRT equ   $E000      ; Cold start BASIC
>44   FIXLINKS equ   $D4F2      ; Fix up BASIC prog links
>45   RUNPROG  equ   $D566      ; RUN Applesoft prog
>46
>47   * Mapping of hardware resources
>48
>49   dsend    equ   an1         ; Data 'send'
>50   drecv    equ   pb1         ; Data 'receive'
>51   zipslow  equ   dsk6off     ; Zip Chip 'slow mode' for 51 ms.
```

```
>53   * Page zero variables
>54
>55   lastidx  equ   $EB        ; Last RCVPKT buffer index
>56   ckbyte   equ   $EC        ; Check byte
>57   ptr      equ   $ED        ; Data buffer pointer (0..leng-1)
>58   address  equ   $FC        ; Scratch addr of local data
>59   length   equ   $FE        ; Scratch length of local data
>60
>61   * Protocol constants
>62
>63   cyperms  equ   1020       ; Cycles per ms. (really 1020.4)
>64
>65   arbtime  equ   1          ; Min arbitration time (ms)
>66   ]cy      equ   arbtime*cyperms ; Arbtime in cycles
>67   ]cpx     equ   11         ; Cycles per X iteration
>68   arbx     equ   ]cy/]cpx   ; X iterations
>69
>70   ]servpad equ   ]cy/4      ; Gap margin
>71   servegap equ   ]cy-]servpad/13 ; SERVER wait loop 13 cyc.
>72
>73   ]cy      equ   ]cpx*256   ; Max arb time (cycles)
>74   maxarb   equ   ]cy+cyperms/cyperms ; ceiling(max arb) (ms)
>75
>76   idletime equ   20         ; Idle polling timeout (ms)
>77                              ; (stay under 51ms for Zip Chip)
>78   reqdur   equ   6          ; Typical req duration (ms)
>79   reqpidle equ   idletime/reqdur ; Requests per idletime
>80
>81   ]cy      equ   idletime*cyperms ; Timeout in cycles
>82   ]cpx     equ   11         ; Cycles per X iteration
>83   ]cpy     equ   ]cpx*256+4 ; Cycles per Y iteration
>84   idleto   equ   ]cy/]cpy+1 ; Number of Y iterations
>85
>86   reqto    equ   1          ; Timeout within protocol is
>87                              ;  minimum arbitration time.
>88   maxgap   equ   87         ; Max intra-pkt gap (cycles)
>89   gapwait  equ   maxgap/13+1 ; MONITOR wait loop is 13 cyc.
>90
>91   reqtime  equ   3000       ; Req response timeout (ms)
>92   rqperiod equ   20         ; Milliseconds between retrys
>93   reqdelay equ   rqperiod-3 ; ARB+SEND+RCV timeout = 3ms.
>94
>95   maxreqrt equ   3          ; Max # of xxxREQ retries
>96   maxretry equ   reqtime/rqperiod/maxreqrt ; # of re-sends
```

```
 45           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
```

```
                 46   loadpnt  equ   $B800      ; Crate start address
                 47            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
B800: 20 00 B8  >82   entry    jsr   *            ; BOOT entry: init and
B803: 20 03 B8  >83   servelp  jsr   *            ;  Run request server
B806: 4C 03 B8  >84            jmp   servelp      ;    forever...
B809: 4C 09 B8  >85   init     jmp   *            ; Initialize and return
B80C: 4C 0C B8  >86   serve    jmp   *            ; Run reqest server
B80F: 4C 0F B8  >87   peek     jmp   *            ; Peek/Poke 'sbuf+dst' for
B812: 4C 12 B8  >88   poke     jmp   *            ;  'sbuf+len' bytes at 'sbuf+adr'
B815: 4C 15 B8  >89   call     jmp   *            ; Call 'sbuf+dst' at 'sbuf+adr'
B818: 4C 18 B8  >90   putmsg   jmp   *            ; Put message to server
B81B: 4C 1B B8  >91   getmsg   jmp   *            ; Get message from server
B81E: 4C 1E B8  >92   bcast    jmp   *            ; Broadcast data
B821: 4C 21 B8  >93   bpoke    jmp   *            ; Broadcast 2-byte POKE
B824: 4C 24 B8  >94   peekinc  jmp   *            ; PEEK & INC 2-byte val
B827: 4C 27 B8  >95   peekpoke jmp   *            ; PEEKPOKE 2-byte val
B82A: 4C 2A B8  >96   run      jmp   *            ; RUN Applseoft prog
B82D: 4C 2D B8  >97   brun     jmp   *            ; BRUN M/L prog
B830: 4C 30 B8  >98   rcvctl   jmp   *            ; Receive control pkt
B833: 4C 33 B8  >99   rcvptr   jmp   *            ; Receive to 'ptr'
B836: 4C 36 B8 >100   rarl=>al jmp   *            ; Rbuf adr,len=>address,length
B839: 4C 39 B8 >101   rcvlong  jmp   *            ; Receive long data
```

```
                    >103  ********** Parameters and variables ************
                    >104
B83C: 00            >105  self     db    0          ; Our own machine ID
B83D: 00 00 00      >106  sbuf     ds    lenctl     ; Control pkt send buffer
B845: 00 00 00      >107  rbuf     ds    lenctl     ; Control pkt receive buffer
B84D: 00 00         >108  locaddr  dw    0          ; Local address of req data
B84F: 00            >109  retrylim db    0          ; Limit of REQUEST resends
B850: 00            >110  servecnt db    0          ; SERVE iterations (0=256)
                    >111
                    >112  parmsiz  equ   *-self     ; Size of parameter area
                    >113
                    >114  ************ Counters and Version **************
                    >115
B851: 00            >116  arbxv    db    0          ; Arbitrate X iters (modified)
B852: 00            >117  tolim    db    0          ; RCVPKT timeout limit
B853: 08            >118  reqctr   db    8          ; SERVER request counter
B854: 00            >119  reqretry db    0          ; xxxREQ retries remaining
B855: 00            >120  retrycnt db    0          ; REQUEST resend count
B856: 00 00         >121  errprot  dw    0          ; Protocol error count
B858: 00 00         >122  ckerr    dw    0          ; Checksum error count
B85A: 00 00         >123  frmcerr  dw    0          ; 'frmc' collision errors
B85C: 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
B85D: 1F            >131  idtable  db    maxid      ; Table of machine attributes
B85E: 00 00 00      >132           ds    maxid      ; Rest of ID table (=0)
                    >133
                    >134           dend
```

```
                        49   ************************************************************
                        50   *                                                          *
                        51   *                      SYNTH.LOADER                        *
                        52   *                                                          *
                        53   ************************************************************
                        54
                        55   * Definitions
                        56
                        57   nav      equ   $7F         ; Next avail page stash
                        58   nleft    equ   $80         ; Voices left and load pages
                        59
                        60   SYNTH    equ   $800        ; SYNTH load address
                        61   startsyn equ   SYNTH+$80   ; SYNTH entry point
                        62   voicetbl equ   SYNTH+$380  ; SYNTH voice table
                        63   synthend equ   SYNTH+$2000 ; Music start address
                        64   music    equ   $08         ; SYNTH music ptr
                        65
0200: 4C 13 02          66   go       jmp   start       ; Jump around data
                        67
0203: 00                68   vpage    db    0*0         ; Next unused page
0204: 00                69   nvoices  db    0*0         ; Number of voices
0205: 00 00 00          70   vlist    ds    14          ; Voice list
                        71
0213: A0 00             72   start    ldy   #signon-msg ; Sign on.
0215: 20 E1 02          73            jsr   prntmsg
0218: AE 04 02          74            ldx   nvoices     ; Set # of voices
021B: 86 80             75            stx   nleft       ;  left to load
021D: A9 00             76            lda   #0          ;    and mark
021F: 95 80             77   :vinit   sta   nleft,x     ;     all unloaded.
0221: CA                78            dex
0222: D0 FB             79            bne   :vinit
0224: 20 D1 02          80   :again   jsr   waitbcst    ; Serve until BCAST
0227: C9 F0             81            cmp   #t_SYNTH    ;  of SYNTH code.
0229: D0 F9             82            bne   :again
                        83            mov16 #SYNTH;address ; Receive SYNTH code
022B: A9 00             83            lda   #SYNTH      ; Move 2 bytes
022D: 85 FC             83            sta   address
022F: A9 08             83            lda   #SYNTH/$100 ; high byte of immediate
0231: 85 FD             83            sta   1+address
                        83            eom
0233: 20 39 B8          84            jsr   rcvlong
0236: B0 EC             85            bcs   :again      ; If NG, try again.
0238: A0 0C             86            ldy   #voices-msg ; Set up for voices.
023A: 20 E1 02          87            jsr   prntmsg
023D: 20 D1 02          88   :vagain  jsr   waitbcst    ; Receive a voice...
0240: C9 F1             89            cmp   #t_VOICE
0242: D0 F9             90            bne   :vagain
0244: AE 04 02          91            ldx   nvoices     ; Do we need this voice?
0247: B5 80             92   :search  lda   nleft,x     ; Valid?
0249: D0 07             93            bne   :skip       ; -No, skip this entry.
024B: BD 04 02          94            lda   nvoices,x   ; -Yes, is voice
024E: C5 FC             95            cmp   address     ;   a hit?
```

```
0250: F0 05     96            beq   :load      ; -Yes, load this voice.
0252: CA        97   :skip    dex               ; Iterate over
0253: D0 F2     98            bne   :search     ;  whole voice table.
0255: F0 E6     99            beq   :vagain     ; Try again. (always)
                100
0257: 18        101  :load    clc               ; Compute ending page
0258: AD 03 02  102           lda   vpage
025B: 65 FF     103           adc   length+1
025D: 85 7F     104           sta   nav         ; Save next avail page
025F: CD CF 03  105           cmp   nadapage    ; Compare to NadaNet
0262: 90 08     106           bcc   :ok         ; -Less is OK
0264: A0 1B     107           ldy   #overflow-msg ; Memory overflow!
0266: 20 E1 02  108           jsr   prntmsg
0269: 4C 03 B8  109           jmp   servelp     ; Stop & serve.
                110
026C: A9 00     111  :ok      lda   #0          ; Set up receive
026E: 85 FC     112           sta   address     ;  address.
0270: AD 03 02  113           lda   vpage
0273: 85 FD     114           sta   address+1
0275: 20 39 B8  115           jsr   rcvlong     ; Receive the voice.
0278: B0 C3     116           bcs   :vagain     ; -Err.  Try again.
027A: A9 A0     117           lda   #"          " ; Print loaded voice #
027C: 20 ED FD  118           jsr   COUT
027F: AD 49 B8  119           lda   rbuf+adr    ; Print in hex.
0282: 20 DA FD  120           jsr   PRBYTE
0285: AE 04 02  121           ldx   nvoices     ; Scan voice list
0288: B5 80     122  :vloop   lda   nleft,x     ;  for matches.
028A: D0 12     123           bne   :no         ; -Already loaded
028C: BD 04 02  124           lda   nvoices,x   ; Check match
028F: CD 49 B8  125           cmp   rbuf+adr    ;  with loaded voice.
0292: D0 0A     126           bne   :no         ; -Nope, try next.
0294: AD 03 02  127           lda   vpage       ; -Match:
0297: 95 80     128           sta   nleft,x     ;   Mark loaded
0299: 9D 7F 0B  129           sta   voicetbl-1,x ; Fix SYNTH's voicetbl
029C: C6 80     130           dec   nleft       ;   One less to load
029E: CA        131  :no      dex               ; Iterate over
029F: D0 E7     132           bne   :vloop      ;  whole voice table.
02A1: AD 03 02  133           lda   vpage       ; Set pointer to
02A4: 85 FD     134           sta   address+1   ;  envelope page.
02A6: A9 00     135           lda   #0
02A8: 85 FC     136           sta   address
02AA: A8        137           tay
02AB: 18        138           clc
02AC: B1 FC     139  :reloc   lda   (address),y ; Relocate envelope
02AE: 65 FD     140           adc   address+1   ;   page pointers.
02B0: 91 FC     141           sta   (address),y
02B2: C8        142           iny
02B3: D0 F7     143           bne   :reloc
02B5: A5 7F     144           lda   nav         ; Recover next avail page
02B7: 8D 03 02  145           sta   vpage
02BA: A5 80     146           lda   nleft       ; All pages loaded?
02BC: F0 03     147           beq   :ready      ; -Yes.
```

```
02BE: 4C 3D 02  148           jmp   :vagain    ; -No, keep loading.
                149
02C1: A0 13     150  :ready   ldy   #ready-msg ; -Yes, announce "Ready"
02C3: 20 E1 02  151           jsr   prntmsg
                152           mov16 #synthend;music ; Set music ptr
02C6: A9 00     152           lda   #synthend  ; Move 2 bytes
02C8: 85 08     152           sta   music
02CA: A9 28     152           lda   #synthend/$100 ; high byte of immediate
02CC: 85 09     152           sta   1+music
                152           eom
02CE: 4C 80 08  153           jmp   startsyn   ;       and start SYNTH.
                154
02D1: 20 0C B8  155  waitbcst jsr   serve      ; Serve requests...
02D4: AD 45 B8  156           lda   rbuf+rqmd  ; Is it a BCAST?
02D7: C9 41     157           cmp   #r_BCAST+rm_REQ
02D9: D0 F6     158           bne   waitbcst   ; -No, keep serving.
02DB: EE 45 B8  159           inc   rbuf+rqmd  ; -Yes. (Just once!)
02DE: A5 FD     160           lda   address+1  ; Load BCAST data tag
02E0: 60        161           rts              ;  and return.
                162
                163  * Message handling
                164
02E1: B9 ED 02  165  prntmsg  lda   msg,y      ; Print message (Y)
02E4: F0 06     166           beq   :done      ; Null terminates.
02E6: 20 ED FD  167           jsr   COUT
02E9: C8        168           iny
02EA: D0 F5     169           bne   prntmsg    ; (always)
                170
02EC: 60        171  :done    rts              ; Return
                172
                173  msg      equ   *          ; Message table origin
02ED: C3 F2 E1  174  signon   asc   "CrateSynth",8D,00
02F9: D6 EF E9  175  voices   asc   "Voices",00
0300: 8D        176  ready    db    $8D
0301: D2 E5 E1  177           asc   "Ready",8D,00
0308: 8D        178  overflow db    $8D
0309: CD E5 ED  179           asc   "Memory overflow",8D,00
                180
                181  endcode  equ   *
                182           err   *-1/$3C0   ; Can't exceed $3C0


--End assembly, 282 bytes, Errors: 0


Symbol table - alphabetical order:

?  BELL    =$FF3A   ?  COLDSTRT=$E000      COUT    =$FDED   ?  CROUT1  =$FD8B
?  CSW     =$36     ?  FIXLINKS=$D4F2   ?  FRETOP  =$6F     ?  HIMEM   =$73
?  HOME    =$FC58   ?  KSW     =$38     ?  ONERR   =$D8     ?  PRBL2   =$F94A
   PRBYTE  =$FDDA   ?  PREAD   =$FB1E   ?  PROGEND =$AF     ?  PSTART  =$67
?  PWREDUP =$03F4   ?  ROMboot =$00     ?  RUNPROG =$D566   ?  SOFTEV  =$03F2
```

```
   SYNTH   =$0800   ?  VARTAB  =$69     ?  VBL     =$C019   V  ]cpx    =$0B
V  ]cpy    =$0B04   V  ]cy     =$4FB0   V  ]servpad=$FF        address =$FC
   adr     =$04     MD?align   =$8000      an0     =$C058      an1     =$C05A
?  an2     =$C05C   ?  an3     =$C05E      arbtime =$01     ?  arbx    =$5C
?  arbxv   =$B851   ?  bcast   =$B81E   ?  bootself=$03CC   ?  bpoke   =$B821
?  brun    =$B82D   ?  call    =$B815   ?  ckbyte  =$EC     ?  ckerr   =$B858
?  crate   =$01        cyperms =$03FC   MD?delay   =$8000   MD?dlyms   =$8000
?  dos     =$00     ?  drecv   =$C062   ?  dsend   =$C05A      dsk6off =$C0E8
?  dst     =$02     ?  endcode =$031A   ?  enhboot =$00     ?  entry   =$B800
?  errprot =$B856   ?  frm     =$03     ?  frmc    =$01     ?  frmcerr =$B85A
?  gapwait =$07     ?  getmsg  =$B81B   ?  go      =$0200      idletime=$14
?  idleto  =$08     ?  idtable =$B85D   MD?inc16   =$8000   ?  init    =$B809
?  kbstrobe=$C010   ?  keybd   =$C000   ?  lastidx =$EB     ?  len     =$06
   lenctl  =$08        length  =$FE        loadpnt =$B800   ?  locaddr =$B84D
?  master  =$00     ?  maxarb  =$03        maxgap  =$57        maxid   =$1F
?  maxreq  =$70        maxreqrt=$03     ?  maxretry=$32     ?  modmask =$07
MD mov16   =$8000   ?  mserve  =$00        msg     =$02ED      music   =$08
   nadapage=$03CF   ?  nadaver =$B85C      nav     =$7F        nleft   =$80
   nvoices =$0204      overflow=$0308   ?  parmsiz =$15        pb0     =$C061
   pb1     =$C062   ?  pb2     =$C063   ?  peek    =$B80F   ?  peekinc =$B824
?  peekpoke=$B827   ?  poke    =$B812      prntmsg =$02E1   ?  ptr     =$ED
?  ptrig   =$C070   ?  putmsg  =$B818      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   =$60     ?  rarl=>al=$B836
   rbuf    =$B845   ?  rcvctl  =$B830      rcvlong =$B839   ?  rcvptr  =$B833
   ready   =$0300   ?  reqctr  =$B853   ?  reqdelay=$11        reqdur  =$06
   reqfac  =$08     ?  reqmask =$F8     ?  reqpidle=$03     ?  reqretry=$B854
   reqtime =$0BB8   ?  reqto   =$01     ?  retrycnt=$B855   ?  retrylim=$B84F
?  rm_ACK  =$02     ?  rm_DACK =$03     ?  rm_NAK  =$04        rm_REQ  =$01
   rqmd    =$00        rqperiod=$14     ?  run     =$B82A   ?  sbuf    =$B83D
   self    =$B83C      serve   =$B80C   ?  servecnt=$B850   ?  servegap=$3A
   servelp =$B803      signon  =$02ED   ?  spkr    =$C030      start   =$0213
   startsyn=$0880      synthend=$2800   ?  t_BASIC =$E0        t_SYNTH =$F0
   t_VOICE =$F1     ?  tolim   =$B852      version =$31     ?  vlist   =$0205
   voices  =$02F9      voicetbl=$0B80      vpage   =$0203      waitbcst=$02D1
?  warmstrt=$03CD   ?  zipslow =$C0E8
```

Symbol table - numerical order:

```
?  master  =$00     ?  dos     =$00     ?  mserve  =$00     ?  ROMboot =$00
?  enhboot =$00        rqmd    =$00     ?  crate   =$01        arbtime =$01
?  reqto   =$01     ?  frmc    =$01        rm_REQ  =$01     ?  dst     =$02
?  rm_ACK  =$02     ?  maxarb  =$03     ?  reqpidle=$03        maxreqrt=$03
?  frm     =$03     ?  rm_DACK =$03        adr     =$04     ?  rm_NAK  =$04
   reqdur  =$06     ?  len     =$06     ?  gapwait =$07     ?  modmask =$07
?  idleto  =$08        lenctl  =$08        reqfac  =$08     ?  r_PEEK  =$08
   music   =$08     V  ]cpx    =$0B     ?  r_POKE  =$10     ?  reqdelay=$11
   idletime=$14        rqperiod=$14     ?  parmsiz =$15     ?  r_CALL  =$18
   maxid   =$1F     ?  r_PUTMSG=$20     ?  r_GETMSG=$28     ?  r_GETID =$30
   version =$31     ?  maxretry=$32     ?  CSW     =$36     ?  KSW     =$38
```

```
?  r_BOOT  =$38     ?  servegap=$3A        r_BCAST =$40     ?  r_BPOKE =$48
?  r_PKINC =$50        maxgap  =$57     ?  r_PKPOK =$58     ?  arbx    =$5C
?  r_RUN   =$60     ?  PSTART  =$67     ?  r_BRUN  =$68     ?  VARTAB  =$69
?  FRETOP  =$6F     ?  maxreq  =$70     ?  HIMEM   =$73        nav     =$7F
   nleft   =$80     ?  PROGEND =$AF     ?  ONERR   =$D8     ?  t_BASIC =$E0
?  lastidx =$EB     ?  ckbyte  =$EC     ?  ptr     =$ED        t_SYNTH =$F0
   t_VOICE =$F1     ?  reqmask =$F8        address =$FC        length  =$FE
V  ]servpad=$FF     ?  go      =$0200      vpage   =$0203      nvoices =$0204
?  vlist   =$0205      start   =$0213      waitbcst=$02D1      prntmsg =$02E1
   msg     =$02ED      signon  =$02ED      voices  =$02F9      ready   =$0300
   overflow=$0308   ?  endcode =$031A   ?  bootself=$03CC   ?  warmstrt=$03CD
   nadapage=$03CF   ?  SOFTEV  =$03F2   ?  PWREDUP =$03F4      cyperms =$03FC
   SYNTH   =$0800      startsyn=$0880   V  ]cpy    =$0B04      voicetbl=$0B80
   reqtime =$0BB8      synthend=$2800   V  ]cy     =$4FB0   MD?align   =$8000
MD?dlyms   =$8000   MD?delay   =$8000   MD mov16   =$8000   MD?inc16   =$8000
   loadpnt =$B800   ?  entry   =$B800      servelp =$B803   ?  init    =$B809
   serve   =$B80C   ?  peek    =$B80F   ?  poke    =$B812   ?  call    =$B815
?  putmsg  =$B818   ?  getmsg  =$B81B   ?  bcast   =$B81E   ?  bpoke   =$B821
?  peekinc =$B824   ?  peekpoke=$B827   ?  run     =$B82A   ?  brun    =$B82D
?  rcvctl  =$B830   ?  rcvptr  =$B833   ?  rarl=>al=$B836      rcvlong =$B839
   self    =$B83C   ?  sbuf    =$B83D      rbuf    =$B845   ?  locaddr =$B84D
?  retrylim=$B84F   ?  servecnt=$B850   ?  arbxv   =$B851   ?  tolim   =$B852
?  reqctr  =$B853   ?  reqretry=$B854   ?  retrycnt=$B855   ?  errprot =$B856
?  ckerr   =$B858   ?  frmcerr =$B85A   ?  nadaver =$B85C   ?  idtable =$B85D
?  keybd   =$C000   ?  kbstrobe=$C010   ?  VBL     =$C019   ?  spkr    =$C030
   an0     =$C058      an1     =$C05A   ?  dsend   =$C05A   ?  an2     =$C05C
?  an3     =$C05E      pb0     =$C061      pb1     =$C062   ?  drecv   =$C062
?  pb2     =$C063   ?  ptrig   =$C070      dsk6off =$C0E8   ?  zipslow =$C0E8
?  FIXLINKS=$D4F2   ?  RUNPROG =$D566   ?  COLDSTRT=$E000   ?  PRBL2   =$F94A
?  PREAD   =$FB1E   ?  HOME    =$FC58   ?  CROUT1  =$FD8B      PRBYTE  =$FDDA
   COUT    =$FDED   ?  BELL    =$FF3A
```