```
1   ************************************************************
2   *                                                          *
3   *              C R A T E . B P R U N N E R                 *
4   *                                                          *
5   *            Michael J. Mahon - May 31, 2008               *
6   *                 Revised Jan 24, 2009                     *
7   *                                                          *
8   *    Copyright (c) 1996, 2003, 2004, 2005, 2008, 2009      *
9   *                                                          *
10  *   CRATE.BPRUNNER uses the BCAST protocol to receive      *
11  *   a BASIC program on all machines on which it is         *
12  *   running.                                               *
13  *                                                          *
14  *   It sets the KSW vector to re-enter SERVELP, so that    *
15  *   the program ends upon its first request for input.     *
16  *                                                          *
17  *   CR.BPRUNNER performs the following actions:            *
18  *     1. Coldstarts Applesoft                              *
19  *     2. Serves a BCAST request with tag $E0xx to          *
20  *        receive the Applesoft program                     *
21  *     3. Sets up Applesoft pointers                        *
22  *     4. RUNs the program                                  *
23  *                                                          *
24  ************************************************************
25  *                                                          *
26  *                    Change History                        *
27  *                                                          *
28  *   01/24/09:                                              *
29  *                                                          *
30  *   Added code to clear ONERR flag before RUN.             *
31  *                                                          *
32  *   10/06/08:                                              *
33  *                                                          *
34  *   Revised to use SERVER to serve BCAST request.          *
35  *                                                          *
36  *   09/26/08:                                              *
37  *                                                          *
38  *   Modified to use NADAUSER definitions.                  *
39  *                                                          *
40  *   08/21/08:                                              *
41  *                                                          *
42  *   Modified to use new BCAST request and v3.0 packet      *
43  *   format.                                                *
44  *                                                          *
45  *   05/31/08:                                              *
46  *                                                          *
47  *   First version, adapted from ROM boot code.             *
48  *                                                          *
49  ************************************************************
```

```
 51   ******************** Version setup ************************
 52
 53            org   $200       ; Load to page 2
 54   master   equ   0          ; Non-master version
 55   dos      equ   0          ; Non-DOS version
 56   crate    equ   1          ; Crate version compatibiluty
 57   mserve   equ   0          ; Non-Message Server version
 58   ROMboot  equ   0          ; ROM boot version
 59   enhboot  equ   0          ; Enhanced //e version
 60
 61           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
```

```
 62           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
```

```
                 63   loadpnt  equ   $B800      ; Crate start address
                 64            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
```

```
                        66   ************************************************************
                        67   *                                                          *
                        68   *                      CRATE.BPRUNNER                      *
                        69   *                                                          *
                        70   ************************************************************
                        71
0200: A0 00             72   BPRUNNER ldy   #0          ; Print the "Awaiting prog"
0202: B9 C0 02          73   :msglp   lda   wtmsg,y     ;  message.
0205: 20 ED FD          74            jsr   COUT
0208: C8                75            iny
0209: C0 21             76            cpy   #]wtlen
020B: 90 F5             77            bcc   :msglp
                        78            mov16 #:resume;KSW ; Set to get control
020D: A9 18             78            lda   #:resume   ; Move 2 bytes
020F: 85 38             78            sta   KSW
0211: A9 02             78            lda   #:resume/$100 ; high byte of immediate
0213: 85 39             78            sta   1+KSW
                        78            eom
0215: 4C 00 E0          79            jmp   COLDSTRT   ;    after coldstart.
                        80   :resume  mov16 #warmstrt;KSW ; First input ends prog.
0218: A9 CD             80            lda   #warmstrt  ; Move 2 bytes
021A: 85 38             80            sta   KSW
021C: A9 03             80            lda   #warmstrt/$100 ; high byte of immediate
021E: 85 39             80            sta   1+KSW
                        80            eom
0220: AD CF 03          81            lda   nadapage   ; Set HIMEM
0223: 85 74             82            sta   HIMEM+1    ;  and FRETOP to
0225: 85 70             83            sta   FRETOP+1   ;   NadaNet load page.
0227: A9 00             84            lda   #0         ; Clear the
0229: 85 D8             85            sta   ONERR      ;  ONERR flag.
022B: 20 0C B8          86   :restart jsr   serve      ; Serve requests...
022E: AD 45 B8          87            lda   rbuf+rqmd  ; Is it a BCAST request?
0231: C9 41             88            cmp   #r_BCAST+rm_REQ
0233: D0 F6             89            bne   :restart   ; -No, serve again.
0235: EE 45 B8          90            inc   rbuf+rqmd  ; -Yes, just once!
0238: A5 FD             91            lda   address+1  ; Is hi byte of tag
023A: C9 E0             92            cmp   #t_BASIC   ;  = BASIC prog?
023C: D0 ED             93            bne   :restart   ; -No, not our BCAST!
                        94            mov16 #$801;address ; Set address to $801
023E: A9 01             94            lda   #$801      ; Move 2 bytes
0240: 85 FC             94            sta   address
0242: A9 08             94            lda   #$801/$100 ; high byte of immediate
0244: 85 FD             94            sta   1+address
                        94            eom
0246: 20 39 B8          95            jsr   rcvlong    ; Read the program.
0249: B0 E0             96            bcs   :restart   ; -error, start over.
024B: A5 67             97            lda   PSTART     ; Set up PROGEND
024D: 6D 4B B8          98            adc   rbuf+len   ;  and VARTAB to
0250: 85 AF             99            sta   PROGEND    ;   end of program.
0252: 85 69             100           sta   VARTAB
0254: A5 68             101           lda   PSTART+1
0256: 6D 4C B8          102           adc   rbuf+len+1
```

```
0259: 85 B0     103           sta   PROGEND+1
025B: 85 6A     104           sta   VARTAB+1
025D: 4C 66 D5  105           jmp   RUNPROG    ; RUN the Applesoft prog.
                106
                107           err   *-1/$280   ; BPRUN &POKEs IDTBL
0260: 00 00 00  108           ds    $2C0-*     ;  into $280..$2BF.
                109
                110  * BPRUNNER message
                111
02C0: C1 F7 E1  112  wtmsg    asc   "Awaiting broadcast BASIC program",8D
                113  ]wtlen   equ   *-wtmsg    ; Length of msg.
                114
                115           align 256        ; Align to page boundary
02E1: 00 00 00  115           ds    *-1/256*256+256-*
                115           eom
                116  endcode  equ   *
                117           err   *-1/$300   ; Can't exceed $300


--End assembly, 256 bytes, Errors: 0


Symbol table - alphabetical order:

?  BELL    =$FF3A   ?  BPRUNNER=$0200      COLDSTRT=$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      VARTAB  =$69     ?  VBL     =$C019   V  ]cpx    =$0B
V  ]cpy    =$0B04   V  ]cy     =$4FB0   V  ]servpad=$FF     V  ]wtlen  =$21
   address =$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 =$0300   ?  enhboot =$00
?  entry   =$B800   ?  errprot =$B856   ?  frm     =$03     ?  frmc    =$01
?  frmcerr =$B85A   ?  gapwait =$07     ?  getmsg  =$B81B      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        nadapage=$03CF   ?  nadaver =$B85C
?  parmsiz =$15        pb0     =$C061      pb1     =$C062   ?  pb2     =$C063
?  peek    =$B80F   ?  peekinc =$B824   ?  peekpoke=$B827   ?  poke    =$B812
?  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  =$B833   ?  reqctr  =$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   ?  spkr    =$C030      t_BASIC =$E0     ?  t_SYNTH =$F0
?  t_VOICE =$F1     ?  tolim   =$B852      version =$31        warmstrt=$03CD
   wtmsg   =$02C0   ?  zipslow =$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
V  ]cpx    =$0B     ?  r_POKE  =$10     ?  reqdelay=$11        idletime=$14
   rqperiod=$14     ?  parmsiz =$15     ?  r_CALL  =$18        maxid   =$1F
?  r_PUTMSG=$20     V  ]wtlen  =$21     ?  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   =$73        PROGEND =$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     ?  BPRUNNER=$0200
   wtmsg   =$02C0   ?  endcode =$0300   ?  bootself=$03CC       warmstrt=$03CD
   nadapage=$03CF   ?  SOFTEV  =$03F2   ?  PWREDUP =$03F4       cyperms =$03FC
V  ]cpy    =$0B04      reqtime =$0BB8   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
```