```
                ;********** MON **********

                ;
                ;               65(C)02 MONITOR
                ;       Written by E.J.M. Visschedijk
                ;     Some parts and improvements by N. de Vries
                ;
                ;Copyright (c) 1986 Kim gebruikersclub Nederland
                ;
                ;
                ;History:
                ;V1.00          dd. ??.??.85
                ;V1.01          dd. ??.??.85
                ;V2.01          dd. ??.03.86    Boot routine for DOS65 V2.01
                ;V2.01b         dd. ??.07.86    Disk version for DOS65 V2.01 with Octopus I/O
                ;V2.02          dd. 28.09.86    Disassembler in AS syntax
                ;V2.03          dd. 28.09.86    T command added, Toggle CPU-type
                ;V2.04          dd. 04.10.86    No zero page usage anymore
                ;V2.05          dd. 22.12.86    W command with replace option
                ;V2.06          dd. 25.12.86    L command with PC disassembly
                ;                               Correct PC display
                ;                               E starts from PC not breakpoint
                ;V2.07          dd. ??.??.87    A command added, one line assembler

        9000    monbeg  equ     $9000           ;Begin monitor
        2E32    ver     equ     ".2"            ;version 2 (NOTE: BACKWARDS!)
        3630    release equ     "60"            ;release number (NOTE: BACKWARDS!)

                ;DOS entries

        C020    input   equ     $c020
        C023    output  equ     $c023
        C02F    crlf    equ     $c02f
        C032    spa     equ     $c032
        C035    hnout   equ     $c035
        C038    prbyte  equ     $c038
        C03B    prtext  equ     $c03b
        C041    loupch  equ     $c041

                ;IO65 definitions

        E770    svaint  equ     $e770
        E7B3    brkvector equ   $e7b3
        E7B1    nmivector equ   $e7b1
        E76E    swivector equ   $e76e

                ;Start variable aria

        9F10            org     monbeg+$f10
9F10            v
9F10            delim   res     1               ;Kind of delimiter between parameters
9F11            c02typ  res     1               ;Select kind of 65C02
9F12            nctype  res     1               ;NMOS/CMOS flag for disassembler
9F13            dislct  res     1               ;Number of lines to disassemble
9F14            dntcr   res     1               ;Don't care character for W command
9F15            savstackp res   1               ;Software stackpointer
9F16            bufpoint res    1               ;Inputbuffer pointer
9F17            svstck  res     1               ;Stackpointer saveaddress in mainprogram
9F18            pntx    res     1               ;Program in ram, self modifying
9F19            pntxl   res     1
9F1A            pntxh   res     2               ;Also one byte for RTS
9F1C            pnty    res     1               ;Program in ram, self modifying
9F1D            pntyl   res     1
9F1E            pntyh   res     2               ;Also one byte for RTS
9F20            paral   res     1               ;Parameter A
9F21            parah   res     1
9F22            parbl   res     1               ;Parameter B
9F23            parbh   res     1
9F24            parcl   res     1               ;Parameter C
9F25            parch   res     1
```

```
9F26                inl     res     1               ;Lowbyte input parameter
9F27                inh     res     1               ;Highbyte input parameter
          9F26  pntxlsave equ   inl                 ;Saveaddress for PNTX
          9F27  pntxhsave equ   inh                 ;during H command
9F28                prbffg  res     1               ;'Print inputbuffer flag' after error
9F29                count   res     1               ;Count number of lines in D command
9F2A                inslen  res     1               ;Instruction length in D command
9F2B                tmp4    res     2               ;Workarea for mnemonic printout
9F2D                cmpsta  res     1               ;Store/Compare flag in C, F, M and V command
          9F2A  eindvlag equ   inslen               ;End of hexdump, errorflag in arithmetics
          9F2A  wslength equ   inslen               ;Length of search value in W
          9F29  wrlength equ   count                ;Length of replace value in W
          9F29  res      equ   count                ;Temporary arithmetic result
          9F2B  delimw   equ   tmp4                 ;Search delimeter in W command
          9F2B  deci1    equ   tmp4                 ;Variables for
          9F2C  deci2    equ   tmp4+1               ;decimal to
          9F2D  deci3    equ   cmpsta               ;hex conversion
          9F2D  aapgew   equ   cmpsta               ;Changed something-flag for @ command
          9F2D  tmp6     equ   cmpsta               ;Addressmode code in D command
                ;schei   equ   count                ;Kind of delimiter
                ;onderg  equ   inslen               ;Low border for get decimal
                ;boveng  res   1                    ;High border for get decimal
9F2E                spuser  res     1               ;Stack pointer register
9F2F                yreg    res     1               ;Y register
9F30                xreg    res     1               ;X register
9F31                acc     res     1               ;A register
9F32                pcl     res     1               ;Program counter low register
9F33                pch     res     1               ;Program counter high register
9F34                preg    res     1               ;Processor status
9F35                whbuf   res     16              ;'Look for' buffer
9F45                wsbuf   res     16              ;'Replace with' buffer
9F55                bkptf1  res     1               ;Breakpoint set flag
9F56                bkptl   res     1               ;Address breakpoint lowbyte
9F57                bkpth   res     1               ;Address breakpoint highbyte
9F58                bkptvr  res     1               ;Data from breakpointaddress
9F59                ndvrs
9F59                prcnt   res     1               ;Number of prompt characters

          9F90          org     monbeg+$f90

9F90 4D4F4E3E20 prompt  fcc     'MON> ',0
9F95 00
9F96                reserve res     11
9FA1                qcomvr  res     1               ;Variable for monitor as subroutine
9FA2                brkvs   res     2               ;Saveaddress BREAK vector
9FA4                nmivs   res     2               ;Saveaddress NMI vector
9FA6                swivs   res     2               ;Saveaddress software interrupt vector
9FA8                tabvec  res     2               ;End of cmd.interpr.-routine vec., 2 bytes

9FAA                userx   res     2               ;Vector for user def. command X
9FAC                usery   res     2               ;Vector for user def. command Y

                ;*** Buffer area ***

          9FB0          org     monbeg+$fb0

9FB0                inpbuffer res   80              ;Input/command buffer, 80 char.

                ;********** MON2 **********

          9000          org     monbeg
                ;
                ;*** Start of program: save and set vectors ***
                ;
9000 08           mon65   php                     ;save processor status on stack
9001 78                   sei                     ;disable external IRQ's
9002 BA                   tsx                     ;get stack pointer
9003 8E A19F              stx     qcomvr          ;into save area
                ;
9006 AD B3E7              lda     brkvector       ;get old BRK-vector
```

```
9009 8D A29F           sta     brkvs           ;into
900C AD B4E7           lda     brkvector+1     ;save area
900F 8D A39F           sta     brkvs+1
9012 A9 8C             lda     #break&255      ;and set
9014 8D B3E7           sta     brkvector       ;new
9017 A9 90             lda     #break>>8       ;BRK-vector
9019 8D B4E7           sta     brkvector+1     ;for monitor
                ;
901C AD 6EE7           lda     swivector       ;get old
901F 8D A69F           sta     swivs           ;software interrupt
9022 AD 6FE7           lda     swivector+1     ;vector
9025 8D A79F           sta     swivs+1         ;into save area
9028 A9 9F             lda     #(swint-1)&255  ;and set
902A 8D 6EE7           sta     swivector       ;new
902D A9 90             lda     #(swint-1)>>8   ;software interrupt vector
902F 8D 6FE7           sta     swivector+1     ;for monitor
                ;
9032 AD B1E7           lda     nmivector       ;save old NMI vector
9035 8D A49F           sta     nmivs           ;into
9038 AD B2E7           lda     nmivector+1     ;save area
903B 8D A59F           sta     nmivs+1
903E A9 A3             lda     #nmi&255        ;and set new NMI vector
9040 8D B1E7           sta     nmivector
9043 A9 90             lda     #nmi>>8
9045 8D B2E7           sta     nmivector+1
                ;
                ;*** Initialise monitor variables ***
                ;
9048 28                plp                     ;regain CPU status
9049 A9 00             lda     #$00            ;clear all variables to $00
904B A2 48             ldx     #(ndvrs-v-1)&255 ;get loopcounter
904D 9D 0F9F    rsta   sta     v-1,x
9050 CA                dex                     ;adapt counter
9051 D0 FA             bne     rsta            ;if not all done, loop
                ;
                ;*** Fill inputbuffer with returns ***
                ;
9053 A2 4F             ldx     #79             ;get counter
9055 A9 0D             lda     #$0D            ;and a CR
9057 9D B09F    rstb   sta     inpbuffer,x     ;fill inputbuffer with it
905A CA                dex                     ;adapt counter
905B 10 FA             bpl     rstb            ;and loop until finished
                ;
                ;*** Initialise non-zero variables ***
                ;
905D A0 00      setvar ldy     #$00            ;get offset zero
905F B9 7390    setvr1 lda     vart1,y         ;get variable offset from table
9062 C9 FF             cmp     #$FF            ;if at end
9064 F0 0A             beq     setvr2          ;start monitor
9066 AA                tax                     ;else get offset into variable table in X
9067 B9 8090           lda     vart2,y         ;get initial value
906A 9D 109F           sta     v,x             ;and initialise variable
906D C8                iny                     ;adapt offset/counter
906E D0 EF             bne     setvr1          ;and loop (branch always)
9070 4C CD90    setvr2 jmp     hoofdprog       ;jump to the main program
                ;
                ;*** Offsettable for variables ***
                ;(changed by ndv)
9073 1E         vart1  fcc     spuser-v
9074 04                fcc     dntcr-v         ;wildcard for W command
9075 98                fcc     tabvec-v
9076 99                fcc     tabvec+1-v
9077 01                fcc     c02typ-v        ;select 65C02 disassembler
9078 02                fcc     nctype-v        ;NMOS/CMOS flag for disassembler
9079 03                fcc     dislct-v        ;number of lines to disassemble
907A 00                fcc     delim-v         ;default delimiter between parameters
907B 9A                fcc     userx-v         ;X command vector to
907C 9B                fcc     userx+1-v       ;user defined routine #1
907D 9C                fcc     usery-v         ;Y command vector to
907E 9D                fcc     usery+1-v       ;user defined routine #2
```

```
907F FF                 fcc     $FF             ;end of table marker
                        ;
                        ;*** Table of initial values for variables ***
                        ;
9080 FF         vart2   fcc     $ff             ;default SPUSER
9081 25                 fcc     '%'             ;default wildcard is percent
9082 0392               fdb     illcoma         ;
9084 01                 fcc     $01             ;default Rockwell disassembler (0 for GTE)
9085 00                 fcc     $00             ;default CMOS disassembler ($40 for NMOS)
9086 14                 fcc     20              ;default line count for disassembler
9087 2C                 fcc     ','             ;DELIM is default a comma
9088 0092               fdb     illcom          ;point X command to illegal command
908A 0092               fdb     illcom          ;point Y command to illegal command
                        ;
                        ;*** Perform break (Action for ^C) ***
                        ;
908C A9 00      break   lda     #$00            ;reset flags
908E 20 3BC0            jsr     prtext          ;print message
9091 0D                 fcc     $0d             ;on new line
9092 1B6E               fcc     $1b,'n'         ;disable inverse
9094 1B47               fcc     $1b,'G'         ;disable graphics
9096 427265616B         fcc     'Break'
909B 0D00               fcc     $0d,0
909D 4C F790            jmp     hfderr          ;then jump to monitor entry
                        ;
                        ;*** Software interrupt routine ***
                        ;
90A0 AD 70E7    swint   lda     svaint
                        ;
                        ;*** NMI service routine ***
                        ;(PC correction added by ndv)
90A3 8D 319F    nmi     sta     acc             ;save Accu,
90A6 8E 309F            stx     xreg            ;X
90A9 8C 2F9F            sty     yreg            ;and Y registers
90AC 68                 pla                     ;get saved flags
90AD 8D 349F            sta     preg            ;into memory
90B0 D8                 cld                     ;clear possible decimal mode
90B1 38                 sec                     ;prepare for subtract
90B2 68                 pla                     ;get saved PC lo
90B3 E9 02              sbc     #2              ;compensate for BRK
90B5 8D 329F            sta     pcl             ;save PC lo in memory
90B8 68                 pla                     ;get PC hi
90B9 E9 00              sbc     #0              ;subtract carry
90BB 8D 339F            sta     pch             ;and complete PC
90BE BA                 tsx                     ;get stack pointer before BRK
90BF 8E 2E9F            stx     spuser          ;save too
90C2 20 5F97            jsr     cmla            ;show all registers
90C5 AD 349F            lda     preg            ;restore old processor status
90C8 48                 pha                     ;through
90C9 28                 plp                     ;stack
90CA 4C F790            jmp     hfderr          ;and finish
                        ;
                        ;*** Main program ***
                        ;
90CD 20 3BC0    hoofdprog jsr   prtext          ;show sign-on message
90D0 0C4D4F4E36         fcc     $0c,'MON65 '
90D5 3520
90D7 322E 3036          fdb     ver,release
90DB 0D1B692048         fcc     '\r',$1b,'i HaViSoft ',$1b,'n\r\n',0
90E0 615669536F
90E5 6674201B6E
90EA 0D0A00
90ED 20 0B95            jsr     showtyp         ;show current CPU-type
90F0 20 2FC0            jsr     crlf            ;start on new line
90F3 BA                 tsx                     ;get stackpointer
90F4 8E 179F            stx     svstck          ;save it
90F7 AE 179F    hfderr  ldx     svstck          ;error entry, so:
90FA 9A                 txs                     ;restore old stack pointer
90FB A2 00      mainpgm ldx     #0              ;get start index
90FD BD 909F    1       lda     prompt,x        ;get prompt character
```

```
9100 F0 06            beq    2.f             ;if end marker, go get input
9102 20 23C0          jsr    output          ;else print prompt
9105 E8               inx                    ;adapt index
9106 D0 F5            bne    1.b             ;and loop (branch always)
9108 8E 599F    2     stx    prcnt           ;then store length of prompt
910B 20 2A91          jsr    vulbuf          ;get input in inputbuffer until CR
910E A9 80            lda    #$80            ;set flag for printing whole inputbuffer
9110 8D 289F          sta    prbffg          ;after error in second command
9113 20 8591    main  jsr    uitbuffer       ;go get command string and execute command
                ;
                ;This is the returnaddress after execution of a command
                ;
9116 20 2FC0          jsr    crlf            ;start on new line
9119 0E 289F          asl    prbffg          ;reset the print input buffer flag
911C AE 169F          ldx    bufpoint        ;get the bufferpointer
911F CA               dex                    ;minus 1
9120 BD B09F          lda    inpbuffer,x     ;get last character of buffer
9123 C9 3A            cmp    #':'            ;if colon
9125 F0 EC            beq    main            ;read and execute next command
9127 4C FB90          jmp    mainpgm         ;else restart
                ;
                ;*** Get input in the input buffer ***
                ;
912A A2 00      vulbuf ldx    #$00            ;reset the counter
912C 8E 169F          stx    bufpoint        ;only for UITBUFFER
912F 20 20C0    bufloop jsr   input           ;get a character
9132 29 7F            and    #$7f            ;filter MSbit off
9134 C9 0D      cr    cmp    #$0d            ;if RETURN
9136 D0 04            bne    1.f
9138 9D B09F    cruit sta    inpbuffer,x     ;move to buffer
913B 60               rts                    ;but do not echo
                ;
913C C9 19      1     cmp    #$19            ;if control Y
913E F0 34            beq    cony            ;go echo previous buffer
9140 C9 20            cmp    #$20            ;other control characters
9142 90 EB            bcc    bufloop         ;are ignored
                ;
9144 C9 7F            cmp    #$7f            ;if DEL/RUB
9146 D0 0F            bne    stabuf
9148 E0 00            cpx    #$00            ;and on 1st buffer position
914A F0 E3            beq    bufloop         ;do not echo
                ;action for DEL/RUB
914C 20 3BC0    delet jsr    prtext          ;else echo BS, space, BS
914F 08200800         fcc    $08,$20,$08,$00 ;to delete character on screen
9153 CA               dex                    ;adapt buffer index
9154 4C 2F91          jmp    bufloop         and loop until CR
                ;other characters
9157 9D B09F    stabuf sta   inpbuffer,x     ;move character in buffer
915A E8               inx                    ;adapt buffer index
915B 20 23C0          jsr    output          ;echo character
915E E0 4F            cpx    #79             ;if end of buffer
9160 D0 CD            bne    bufloop
9162 A9 07      2     lda    #$07            ;get BELL character
9164 20 23C0          jsr    output          ;and ring the bell
9167 20 20C0          jsr    input           ;get next character
916A C9 7F            cmp    #$7f            ;if DEL/RUB
916C F0 DE            beq    delet           ;go do action for DEL/RUB
916E C9 0D            cmp    #$0d            ;if RETURN
9170 F0 C6            beq    cruit           ;finish input
9172 D0 EE            bne    2.b             ;else ignore others (branch always)
                ;action for control-Y
9174 BD B09F    cony  lda    inpbuffer,x     ;get old buffer character
9177 C9 0D            cmp    #$0d            ;if at end of buffer
9179 F0 B4            beq    bufloop         ;go enter input loop
917B 20 23C0          jsr    output          ;else echo character
917E E8               inx                    ;point to next
917F E0 4F            cpx    #79             ;if not at maximum
9181 D0 F1            bne    cony            ;get next old character
9183 F0 AA            beq    bufloop         ;else get input
                ;
```

```
                ;*** Read command from buffer ***
                ;
9185 AE 169F    uitbuffer ldx   bufpoint        ;get the buffer pointer
9188 D0 07              bne     1.f             ;if no input
918A BD B09F            lda     inpbuffer,x     ;get beginning
918D C9 0D              cmp     #$0d            ;if RETURN only
918F F0 23              beq     3.f             ;do nothing
9191 BD B09F    1       lda     inpbuffer,x     ;else get character
9194 20 41C0            jsr     loupch          ;convert lower case to upper
9197 A0 00              ldy     #$00            ;get initial command table index
9199 D9 B591    looktab cmp     comtab,y        ;test if valid command
919C F0 08              beq     2.f             ;if so, go execute command
919E C8                 iny                     ;else try next command
919F C0 17              cpy     #(commtable-comtab) ;until table exhausted
91A1 D0 F6              bne     looktab         ;if no, loop
91A3 6C A89F    error1  jmp     [tabvec]        ;if yes, jump to vector
                ;
91A6 EE 169F    2       inc     bufpoint        ;command found, point to next character
91A9 98                 tya                     ;get table offset in A
91AA 0A                 asla                    ;calculate jumptable offset
91AB AA                 tax                     ;in X
91AC BD CD91            lda     commtable+$01,x ;get highbyte
91AF 48                 pha                     ;on stack
91B0 BD CC91            lda     commtable,x     ;get low as well
91B3 48                 pha
91B4 60         3       rts                     ;and execute command
                ;NOTE: Return address is in mainprogram
                ;
                ;*** Valid commands ***
                ;
91B5 23         comtab  fcc     '#'             ;Calculate from hex. to dec.
91B6 24                 fcc     '$'             ;Calculate from dec. to hex.
91B7 2B                 fcc     '+'             ;Add two words of two bytes
91B8 2D                 fcc     '-'             ;Subtract two words of two bytes
91B9 3C                 fcc     '<'             ;Repeat inputbuffer
91BA 40                 fcc     '@'             ;Get address and change contents
91BB 41                 fcc     'A'             ;One line assembler
91BC 42                 fcc     'B'             ;Breakpoints
91BD 43                 fcc     'C'             ;Check
91BE 44                 fcc     'D'             ;Disassembler
91BF 45                 fcc     'E'             ;Execute
91C0 46                 fcc     'F'             ;Fill
91C1 48                 fcc     'H'             ;Hex and ASCII dump
91C2 4C                 fcc     'L'             ;List cpu registers
91C3 4D                 fcc     'M'             ;Move bytes
91C4 50                 fcc     'P'             ;Clear screen
91C5 51                 fcc     'Q'             ;Quit monitor as a subroutine
91C6 53                 fcc     'S'             ;Compute the checksum
91C7 54                 fcc     'T'             ;Toggle CPU-type
91C8 56                 fcc     'V'             ;Verify bytes
91C9 57                 fcc     'W'             ;Search
91CA 58                 fcc     'X'             ;User defined routine
91CB 59                 fcc     'Y'             ;User defined routine
                ;
                ;*** Table of command addresses ***
                ;
91CC 0B99       commtable fdb   cm23-$01
91CE E698               fdb     cm24-$01
91D0 3199               fdb     cm2b-$01
91D2 5A99               fdb     cm2d-$01
91D4 5197               fdb     cm.-$01
91D6 7295               fdb     cmaap-$01
91D8 0B9F               fdb     cma-$01
91DA 0F98               fdb     cmb-$01
91DC F096               fdb     cmc-$01
91DE 7B9B               fdb     cmd-$01
91E0 D197               fdb     cme-$01
91E2 F796               fdb     cmf-$01
91E4 3D94               fdb     cmh-$01
91E6 5B97               fdb     cml-$01
```

```
91E8 2F96               fdb     cmm-$01
91EA E594               fdb     cmp-$01
91EC 4295               fdb     cmq-$01
91EE C398               fdb     cms-$01
91F0 ED94               fdb     cmt-$01
91F2 2596               fdb     cmv-$01
91F4 729A               fdb     cmw-$01
91F6 F991               fdb     cmx-$01
91F8 FC91               fdb     cmy-$01
                        ;
                        ;*** User command X ***
                        ;
91FA 6C AA9F    cmx     jmp     [userx]         ;jump to user defined command X
                        ;
                        ;*** User command Y ***
                        ;
91FD 6C AC9F    cmy     jmp     [usery]         ;jump to user defined command Y
                        ;
                        ;*** Illegal command found ***
                        ;
9200 CE 169F    illcom  dec     bufpoint        ;get buffer index to command
9203 20 4592    illcoma jsr     prbufpijl       ;print the buffer with arrow
9206 20 3BC0            jsr     prtext          ;print text
9209 070D496C6C         fcc     $07,'\rIllegal command\r\n',0
920E 6567616C20
9213 636F6D6D61
9218 6E640D0A00
921D 4C F790            jmp     hfderr          ;and jump to error entry
                        ;
                        ;*** Illegal parameter(s) found ***
                        ;
9220 CE 169F    illpar  dec     bufpoint        ;get buffer index to parameter
9223 20 4592    illpara jsr     prbufpijl       ;print the buffer with arrow
9226 20 3BC0    illparb jsr     prtext          ;print text
9229 070D496C6C         fcc     $07,'\rIllegal parameter(s)\r\n',0
922E 6567616C20
9233 706172616D
9238 6574657228
923D 73290D0A00
9242 4C F790            jmp     hfderr          ;and go do to error handling
                        ;
                        ;*** Print inputbuffer and arrow to error ***
                        ;
9245 20 2FC0    prbufpijl jsr   crlf            ;start on new line
9248 2C 289F            bit     prbffg          ;if it was the first command
924B 30 12              bmi     illcm2          ;don't print buffer
924D A2 00              ldx     #$00            ;else echo buffer
924F BD B09F    1       lda     inpbuffer,x     ;get buffer character
9252 C9 0D              cmp     #$0d            ;if it is RETURN
9254 F0 06              beq     illcom2         ;go print arrow
9256 20 23C0            jsr     output          ;else echo the character
9259 E8                 inx                     ;adapt buffer pointer
925A D0 F3              bne     1.b             ;and loop (branch always)
                        ;
925C 20 2FC0    illcom2 jsr     crlf            ;start on new line
925F AE 169F    illcm2  ldx     bufpoint        ;get buffer pointer
9262 F0 06              beq     pijl            ;if at beginning, go test for 1st command
9264 20 32C0    2       jsr     spa             ;else print a space
9267 CA                 dex                     ;adapt buffer index
9268 D0 FA              bne     2.b             ;loop until at error
926A 2C 289F    pijl    bit     prbffg          ;if it was the first command
926D 10 06              bpl     1.f
926F AE 599F            ldx     prcnt           ;get prompt length
9272 20 939D            jsr     xspace          ;and print so many spaces
9275 A9 5E      1       lda     #'^'            ;get arrow character
9277 4C 23C0            jmp     output          ;and display error position
                        ;
                        ;*** Place arrow on previous delimiter and display error ***
                        ;
927A 20 8092    zpijlsp jsr     zkvsp           ;look for the previous delimiter
```

```
927D 4C 2392          jmp     illpara         ;and display error message
                      ;
                      ;*** Look for previous space ***
                      ;
9280 AE 169F    zkvsp ldx     bufpoint        ;get pointer
9283 CA         zoek  dex                     ;one place backwards
9284 BD B09F          lda     inpbuffer,x     ;get input character
9287 CD 109F          cmp     delim           ;check if delimiter
928A D0 F7            bne     zoek            ;if not, loop
928C 8E 169F          stx     bufpoint        ;else this is the spot
928F 60               rts                     ;so exit

                      ;********** MON3 **********
                      ;
                      ;*** Get character from inputbuffer, convert lower to upper case ***
                      ;
9290 20 9692    gchb  jsr     gchb1           ;get input character
9293 4C 41C0          jmp     loupch          ;and convert lower to upper case
                      ;
                      ;*** Get character from inputbuffer ***
                      ;changed from X indexed to Y indexed
9296 AC 169F    gchb1 ldy     bufpoint        ;get buffer index
9299 B9 B09F          lda     inpbuffer,y     ;get input character
929C EE 169F          inc     bufpoint        ;adapt buffer pointer
929F 60               rts
                      ;
                      ;*** Get three parameters from inputbuffer
                      ;(eenpar and tweepar integrated by ndv)
92A0 20 FE92    driepar jsr   in              ;get parameter
92A3 CD 109F          cmp     delim           ;check for delimiter
92A6 D0 23            bne     parer           ;if incorrect, reflect error
92A8 20 0A93          jsr     copya           ;else copy first parameter
92AB E8               inx                     ;adapt
92AC E8               inx                     ;parameter index
92AD 2C               fcc     $2c             ;skip next instruction
                      ;
                      ;*** Get two parameters from inputbuffer
                      ;
92AE A2 00      tweepar ldx   #0              ;point to PARA
92B0 20 FE92          jsr     in              ;get next parameter
92B3 CD 109F          cmp     delim           ;check for delimiter
92B6 D0 13            bne     parer           ;if incorrect, reflect error
92B8 20 0F93          jsr     copy            ;else copy parameter
92BB E8               inx                     ;adapt
92BC E8               inx                     ;parameter index
92BD 2C               fcc     $2c             ;skip next instruction
                      ;
                      ;*** Get one parameter from input buffer
                      ;
92BE A2 00      eenpar  ldx   #0              ;point to PARA
92C0 20 1C93          jsr     insch           ;get next parameter
92C3 D0 06            bne     parer           ;exit with N=1 if error occured
92C5 20 0F93          jsr     copy            ;else copy parameter to PAR?
92C8 A0 00            ldy     #$00            ;N=0 for normal exit
92CA 60               rts
                      ;
92CB A0 FF      parer   ldy   #$ff            ;N=1
92CD 60               rts
                      ;
                      ;*** Get three parameters and check A<B ***
                      ;(shortened by ndv)
92CE 20 0194    beg3    jsr   testsp          ;test if next is a space
92D1 20 A092            jsr   driepar         ;if so, get 3 parameters
92D4 10 1C              bpl   beg3a           ;if no error, go check A<B
92D6 4C 2092    beg3err jmp   illpar          ;else report illegal parameter
                      ;
                      ;*** Get two parameters and check A<B ***
                      ;
92D9 20 0194    twpaby  jsr   testsp          ;test for a space
92DC 20 FE92            jsr   in              ;get one parameter
```

```
92DF CD 109F            cmp     delim           ;check for correct delimiter
92E2 D0 F2              bne     beg3err         ;if none, report error
92E4 20 0A93            jsr     copya           ;else copy parameter in PARA
92E7 20 FE92            jsr     in              ;get next parameter
92EA CD 109F            cmp     delim           ;check for correct delimiter
92ED D0 E7              bne     beg3err         ;if none go report error
92EF 20 0D93            jsr     copyb           ;else copy parameter in PARB
92F2 20 2693    beg3a   jsr     chck            ;test if A<B
92F5 B0 06              bcs     twpend          ;exit if so
92F7 20 8092            jsr     zkvsp           ;else get error position in command line
92FA 4C 7A92            jmp     zpijlsp         ;and show error message
92FD 60         twpend  rts
                ;
                ;*** Get parameter ***
                ;
92FE 20 5D93    in      jsr     resin           ;clear result area
9301 20 9092    ina     jsr     gchb            ;get character
9304 20 6693    inb     jsr     hexnum          ;convert from ASCII to hex
9307 10 F8              bpl     ina             ;if last character was hex, loop
9309 60                 rts                     ;else exit with char. after parameter in A
                ;
                ;*** Copy parameter ***
                ;(shortened by ndv)
930A A2 00      copya   ldx     #0              ;point to PARA
930C 2C                 fcc     $2c             ;skip next instruction
930D A2 02      copyb   ldx     #2              ;point to PARB
930F AD 269F    copy    lda     inl             ;get bufferbyte low
9312 9D 209F            sta     paral,x         ;copy to parameter pointed by X
9315 AD 279F            lda     inh             ;get bufferbyte high
9318 9D 219F            sta     parah,x         ;copy to parameter pointed by X
931B 60                 rts                     ;and exit
                ;
931C 20 FE92    insch   jsr     in              ;Get parameter and:
                ;
                ;*** Check for delimiter ***
                ;
931F C9 3A      schtt   cmp     #':'            ;end delimiter must be colon
9321 F0 02              beq     schttend
9323 C9 0D              cmp     #$0d            ;or a RETURN
9325 60         schttend rts                    ;Z=1 if one of these
                ;
                ;*** Check parameters ***
                ;
9326 AD 229F    chck    lda     parbl           ;Subtract B from A
9329 CD 209F            cmp     paral
932C AD 239F            lda     parbh
932F ED 219F            sbc     parah           ;Result is in carry
9332 60                 rts                     ;Error if C=0
                ;
                ;*** Check for end ***
                ;
9333 A0 01      checkend ldy    #1              ;Loopcounter & pointer
9335 B9 229F    check   lda     parbl,y         ;Get byte
9338 D9 199F            cmp     pntxl,y         ;Check if equal
933B D0 05              bne     1.f             ;Branch if not equal
933D 88                 dey                     ;Update pointer
933E 10 F5              bpl     check           ;Next please
9340 18                 clc                     ;C=0 then end reached!!!
9341 60                 rts
9342 38         1       sec                     ;C=1 then not ready yet
9343 60                 rts
                ;
                ;*** Copy parameters ***
                ;(changed by ndv)
9344 AD 249F    copypar lda     parcl           ;PARC to Y
9347 AE 259F            ldx     parch
934A 8D 1D9F            sta     pntyl
934D 8E 1E9F            stx     pntyh
9350 AD 209F            lda     paral
9353 AE 219F            ldx     parah           ;PARA to X
```

```
9356 8D 199F          sta     pntxl
9359 8E 1A9F          stx     pntxh
935C 60               rts
                      ;
                      ;*** Clear inl/inh ***
                      ;
935D A0 00      resin ldy     #$00            ;clear
935F 8C 269F          sty     inl             ;both lo
9362 8C 279F          sty     inh             ;and hi bytes
9365 60               rts                     ;then exit
                      ;
                      ;*** Convert from ASCII to hex ***
                      ;
9366 20 8293    hexnum jsr    ashett          ;ASCII to hex
9369 30 14            bmi     hnub            ;if invalid character, show error
936B A0 04            ldy     #4              ;else get bit counter
936D 0E 269F    hnua  asl     inl             ;Shift INL in INH via carry
9370 2E 279F          rol     inh             ;Into INH
9373 88               dey                     ;4 bits in one nibble
9374 D0 F7            bne     hnua            ;Ready?
9376 0D 269F          ora     inl             ;4 LSB's from accu in INL
9379 8D 269F          sta     inl             ;Save it
937C A0 00            ldy     #0
937E 60               rts                     ;Normal exit, Z=1
937F A0 FF      hnub  ldy     #$ff            ;N=1 then error
9381 60               rts
                      ;
9382 C9 30      ashett cmp    #'0'            ;character must
9384 90 0C            bcc     notvat          ;be a
9386 C9 3A            cmp     #':'            ;number
9388 90 0D            bcc     valt            ;or
938A C9 41            cmp     #'A'            ;a letter
938C 90 04            bcc     notvat          ;between
938E C9 47            cmp     #('F'+1)        ;A and F
9390 90 03            bcc     valt1           ;if not, then
9392 A0 FF      notvat ldy    #$ff            ;N=1
9394 60               rts                     ;and exit
9395 69 09      valt1 adc     #$09            ;add 9 for A...F
9397 29 0F      valt  and     #$0f            ;and remove dirt
9399 60               rts
                      ;
                      ;*** Print a byte and a space ***
                      ;
939A 20 38C0    prbyts jsr    prbyte          ;print a byte
939D 4C 32C0          jmp     spa             ;followed by a space
                      ;
                      ;*** Increment X pointer ***
                      ;
93A0 A2 00      pntxinc ldx   #0              ;point to X
93A2 2C               fcc     $2c             ;skip next instruction
                      ;
                      ;*** Increment Y pointer ***
                      ;
93A3 A2 04      pntyinc ldx   #4              ;point to Y
93A5 FE 199F          inc     pntxl,x         ;increment lo
93A8 D0 03            bne     pntxinca        ;page crossed
93AA FE 1A9F          inc     pntxh,x         ;adapt hi
93AD 60         pntxinca rts                  ;then exit
                      ;
                      ;*** Decrement X pointer ***
                      ;
93AE A2 00      pointdec ldx  #0              ;point to X
93B0 2C               fcc     $2c             ;Skip next instruction
                      ;
                      ;*** Decrement Y pointer ***
                      ;
93B1 A2 04      pntydec ldx   #$04            ;point to Y
93B3 38               sec                     ;prepare for subtract
93B4 BD 199F          lda     pntxl,x         ;get lo
93B7 E9 01            sbc     #1              ;decrement
```

```
93B9 9D 199F          sta     pntxl,x         ;and store lo
93BC B0 03            bcs     pntdce          ;if page crossed
93BE DE 1A9F          dec     pntxh,x         ;adapt hi
93C1 60         pntdce rts                    ;and exit
                ;
                ;*** Print pntx and data in A in hex ***
                ;
93C2 AE 1A9F    prpntxa ldx   pntxh           ;get hi
93C5 AC 199F          ldy     pntxl           ;and lo, then:
                ;
                ;*** Print address in X,Y and data in A ***
                ;
93C8 48         pradda  pha                   ;save A
93C9 20 D693          jsr     pradd           ;print address in XY
93CC 68               pla                     ;restore A
93CD 4C 38C0          jmp     prbyte          ;and print in hex, then exit
                ;
                ;*** Print pntx in hex ***
                ;
93D0 AE 1A9F    prpntx  ldx   pntxh           ;get hi
93D3 AC 199F          ldy     pntxl           ;and lo, then:
                ;
                ;*** Print the address in XY in hex ***
                ;
93D6 8A         pradd   txa                   ;get hi
93D7 20 38C0          jsr     prbyte          ;print in hex
93DA 98               tya                     ;get lo
93DB 4C 9A93          jmp     prbyts          ;print in hex, plus a space
                ;
                ;*** Print CRLF and INH and INL as hex address ***
                ;
93DE 20 2FC0    prhlb2  jsr   crlf            ;start on new line, then:
                ;
                ;*** Print INH and INL as hex address ***
                ;
93E1 AE 279F    prin    ldx   inh             ;get hi
93E4 AC 269F          ldy     inl             ;and lo
93E7 4C D693          jmp     pradd           ;and print in hex
                ;
                ;********** MON4 **********
                ;
                ;*** Common start for several commands without parameters ***
                ;(added by ndv, called by commands L, P, Q and T)
93EA 20 9092    cmdonly jsr   gchb            ;check for correct delimiter
93ED 20 1F93          jsr     schtt
93F0 D0 16            bne     nolttro         ;if not OK, report error
93F2 60         testspa rts                   ;else exit
                ;
                ;*** Common start for several commands with two parameters ***
                ;
93F3 20 0B94    begin   jsr   bgncm2          ;input two hex numbers
93F6 20 2693          jsr     chck            ;PARA<PARB?
93F9 B0 03            bcs     beginc          ;if not,
93FB 4C 7A92          jmp     zpijlsp         ;show arrow and error message
93FE 4C 4493    beginc  jmp   copypar         ;else move paramaters to RAM programs
                ;
                ;*** Check for a space after a command ***
                ;
9401 20 9092    testsp  jsr   gchb            ;get character
9404 C9 20            cmp     #$20            ;if space,
9406 F0 EA            beq     testspa         ;go exit
9408 4C 0092    nolttro jmp   illcom          ;else report error
                ;
                ;*** Input two hex numbers ***
                ;
940B 20 0194    bgncm2  jsr   testsp          ;test for a space
940E 20 AE92          jsr     tweepar         ;get the two parameters
9411 10 DF            bpl     testspa         ;if no error, exit
9413 4C 2092          jmp     illpar          ;display it
                ;
```

```
                ;*** Make selfmodifying routines ***
                ;
9416 A9 AD      adpntx  lda     #$ad            ;LDA ABS
9418 2C                 fcc     $2c             ;skip next instruction
9419 A9 BD      bdpntx  lda     #$bd            ;LDA ABS,X
941B 2C                 fcc     $2c             ;skip next instruction
941C A9 B9      b9pntx  lda     #$b9            ;LDA ABS,Y
941E 2C                 fcc     $2c             ;skip next instruction
941F A9 CD      cdpntx  lda     #$cd            ;CMP ABS
9421 2C                 fcc     $2c             ;skip next instruction
9422 A9 8D      d8pntx  lda     #$8d            ;STA ABS
9424 2C                 fcc     $2c             ;skip next instruction
9425 A9 9D      d9pntx  lda     #$9d            ;STA ABS,X
9427 8D 189F            sta     pntx            ;In program with pointer X
942A A9 60              lda     #$60            ;RTS
942C 8D 1B9F            sta     pntx+$03
942F 60                 rts
                ;
9430 A9 AD      adpnty  lda     #$ad            ;LDA ABS
9432 2C                 fcc     $2c             ;skip next instruction
9433 A9 8D      d8pnty  lda     #$8d            ;STA ABS
9435 8D 1C9F            sta     pnty            ;In program with pointer Y
9438 A9 60              lda     #$60            ;RTS
943A 8D 1F9F            sta     pnty+$03
943D 60                 rts
                ;
                ;*** Command H: Hex and ASCII dump ***
                ;(shortened by ndv)
943E 20 F393    cmh     jsr     begin           ;get parameters and check them
9441 20 1994            jsr     bdpntx          ;make a selfmodifying program
9444 A9 FF              lda     #$ff            ;reset endflag
9446 8D 2A9F            sta     eindvlag
9449 20 2FC0            jsr     crlf            ;print a CRLF
                ;print header
944C A2 08              ldx     #8              ;print
944E 20 939D            jsr     xspace          ;7 spaces
9451 A0 00              ldy     #$00            ;reset columncounter
9453 98         hexdd   tya                     ;get count in A
9454 20 35C0            jsr     hnout           ;print as hex nibble
9457 20 919D            jsr     twspace         ;followed by two spaces
945A C8                 iny                     ;increment column number
945B C0 10              cpy     #$10            ;if not at end
945D D0 F4              bne     hexdd           ;loop, else:
                ;loop entry: test if at end address
945F 20 2FC0    hexdeq  jsr     crlf            ;start on new line
9462 2C 2A9F            bit     eindvlag        ;if at end
9465 30 01              bmi     hexdeaq
9467 60                 rts                     ;exit command
                ;
9468 20 D093    hexdeaq jsr     prpntx          ;print current address in hex
946B 20 3BC0            jsr     prtext          ;complete start of line
946E 3A2000             fcc     ': ',0
9471 A2 00              ldx     #0              ;get initial index
9473 20 189F    hexdf   jsr     pntx            ;get data
9476 20 9A93            jsr     prbyts          ;print in hex followed by a space
                ;
9479 18                 clc                     ;calculate endvalue
947A 8A                 txa                     ;get offset
947B 6D 199F            adc     pntxl           ;Add the the address lo
947E 8D 269F            sta     pntxlsave       ;save the result
9481 AD 1A9F            lda     pntxh           ;get address hi
9484 69 00              adc     #$00            ;add carry
9486 8D 279F            sta     pntxhsave       ;and save too
                ;test if at end address
9489 A0 01              ldy     #$01            ;check if at end
948B B9 229F    hexdg   lda     parbl,y         ;get end parameter
948E D9 269F            cmp     pntxlsave,y     ;compare with current position
9491 D0 08              bne     hexdh           ;branch if not equal
9493 88                 dey                     ;compare also lowbytes
9494 10 F5              bpl     hexdg           ;continue elsewhere
```

```
9496 A9 00            lda     #$00            ;end reached
9498 8D 2A9F          sta     eindvlag        ;set the endflag
949B E8       hexdh   inx
949C E0 10            cpx     #$10            ;on end of line?
949E D0 D3            bne     hexdf           ;branch if not ready
94A0 20 32C0          jsr     spa             ;else print a space
              ;display ASCII part
94A3 A2 00            ldx     #0              ;get initial index again
94A5 20 189F  hexdi   jsr     pntx            ;get byte
94A8 20 B694          jsr     prgrnr          ;convert to print format and print
94AB E8       hexdl   inx                     ;adapt index
94AC E0 10            cpx     #$10            ;at end of line?
94AE D0 F5            bne     hexdi           ;if not, loop
94B0 20 D794          jsr     incpnt          ;increase the pointer by $10
94B3 4C 5F94          jmp     hexdeq          ;jump back in loop
              ;
94B6 C9 20    prgrnr  cmp     #$20
94B8 90 03            bcc     prgrna          ;cranch if control char.
94BA 4C 23C0          jmp     output          ;print the byte in ASCII
94BD 48       prgrna  pha
94BE 20 CA94          jsr     prgrf           ;set graphics
94C1 68               pla
94C2 09 60            ora     #$60            ;otherwise as control char. printed
94C4 20 23C0          jsr     output          ;print graphics character, and:
              ;
              ;*** set various display attributes ***
              ;(rewritten by ndv)
94C7 A9 47    prgrg   lda     #'G'            ;get code for no graphics
94C9 2C               fcc     $2c             ;skip next instruction
94CA A9 46    prgrf   lda     #'F'            ;get code for graphics
94CC A0 1B    presc   ldy     #$1b            ;get an ESC, and:
              ;
              ;*** print the characters in Y and A ***
              ;
94CE 48       xaout   pha                     ;save 2nd character
94CF 98               tya                     ;get first in A
94D0 20 23C0          jsr     output          ;print it
94D3 68               pla                     ;get 2nd back
94D4 4C 23C0          jmp     output          ;prrint as well and exit
              ;
              ;*** Increase X pointers with $10 ***
              ;(shortened by ndv)
94D7 18       incpnt  clc                     ;prepare for add
94D8 AD 199F          lda     pntxl           ;get lo
94DB 69 10            adc     #16             ;16 bytes for 1 line
94DD 8D 199F          sta     pntxl           ;save lo
94E0 90 03            bcc     1.f             ;if page crossed
94E2 EE 1A9F          inc     pntxh           ;adapt hi
94E5 60       1       rts                     ;and exit
              ;
              ;*** Command P: clear screen ***
              ;(shortened by ndv)
94E6 20 EA93  cmp     jsr     cmdonly         ;test if command letter only
94E9 A9 0C            lda     #$0c            ;get a FormFeed
94EB 4C 23C0          jmp     output          ;print it and exit
              ;
              ;*** Command T: toggle CPU-type ***
              ;(added by ndv)
94EE 20 EA93  cmt     jsr     cmdonly         ;test if command letter only
94F1 AD 119F          lda     c02typ          ;get current type
94F4 D0 06            bne     tonmos          ;if Rockwell now, switch to NMOS
94F6 2C 129F          bit     nctype          ;if CMOS now
94F9 50 08            bvc     torock          ;switch to Rockwell
94FB 2C               fcc     $2c             ;else type is NMOS, switch to CMOS (skip next)
94FC A9 40    tonmos  lda     #$40            ;get NMOS flag
94FE 8D 129F          sta     nctype          ;set NMOS/CMOS flag
9501 A9 01            lda     #%00000001      ;and set X7/XF to Standard
9503 49 01    torock  eor     #%00000001      ;toggle CMOS version
9505 8D 119F          sta     c02typ          ;set new type
9508 20 2FC0          jsr     crlf            ;start on next line, then:
```

```
                        ;
                        ;*** Print current CPU-type ***
                        ;(added by ndv)
950B 48         showtyp pha                     ;save A
950C AD 119F            lda     c02typ          ;get processorversion
950F D0 16              bne     rock            ;if Rockwell go print message
9511 20 3BC0            jsr     prtext          ;else print standard banner
9514 5374616E64         fcc     'Standard 65',0
9519 6172642036
951E 3500
9520 2C 129F            bit     nctype          ;if standard CMOS
9523 50 11              bvc     cmoscpu         ;go print C02
9525 70 14              bvs     nmoscpu         ;else print 02
9527 20 3BC0    rock    jsr     prtext          ;Rockwell, print banner
952A 526F636B77         fcc     'Rockwell 65',0
952F 656C6C2036
9534 3500
9536 A9 43      cmoscpu lda     #'C'            ;get a C
9538 20 23C0            jsr     output          ;print it
953B 20 3BC0    nmoscpu jsr     prtext          ;then complete type number
953E 303200             fcc     '02',0
9541 68                 pla                     ;restore A
9542 60                 rts
                        ;
                        ;*** Command Q: exit monitor as a subroutine ***
                        ;(shortened by ndv)
9543 20 EA93    cmq     jsr     cmdonly         ;test if command letter only
9546 08                 php                     ;save flags
9547 78                 sei                     ;inhibit IRQ's
                        ;restore original vectors
9548 AD A29F            lda     brkvs           ;get original
954B 8D B3E7            sta     brkvector       ;BRK-vector
954E AD A39F            lda     brkvs+1         ;back
9551 8D B4E7            sta     brkvector+1     ;into place
                        ;
9554 AD A49F            lda     nmivs           ;do the
9557 8D B1E7            sta     nmivector       ;same for
955A AD A59F            lda     nmivs+1         ;the NMI-vector
955D 8D B2E7            sta     nmivector+1
                        ;
9560 AD A69F            lda     swivs           ;and the
9563 8D 6EE7            sta     swivector       ;software
9566 AD A79F            lda     swivs+1         ;interrupt
9569 8D 6FE7            sta     swivector+1     ;vector
                        ;
956C 20 2FC0            jsr     crlf            ;start on a new line
956F 28                 plp                     ;restore the flags (enable IRQ's)
9570 68                 pla                     ;skip
9571 68                 pla                     ;return address of command Q
9572 60                 rts                     ;and exit to monitor caller
                        ;
                        ;*** Command @: Fetch address and change data ***
                        ;(shortened by ndv)
9573 20 5D93    cmaap   jsr     resin           ;reset input buffers
9576 8C 2D9F            sty     aapgew          ;also reset address changed flag
9579 20 0194            jsr     testsp          ;next character must be a space
957C 20 1694            jsr     adpntx          ;make a ramprog.
957F 20 BE92            jsr     eenpar          ;get one parameter
9582 10 03              bpl     monb            ;if error
9584 4C 2092            jmp     illpar          ;report illegal parameter
9587 CE 169F    monb    dec     bufpoint        ;else point to previous character
958A 20 9092            jsr     qchb            ;get it
958D C9 0D              cmp     #$0d            ;if RETURN
958F F0 03              beq     monc            ;go proceed
9591 4C 0092            jmp     illcom          ;else report illegal command
9594 20 4493    monc    jsr     copypar         ;copy address into RAM program
9597 20 2FC0    monca   jsr     crlf            ;start on new line
959A 20 5D93            jsr     resin           ;clear input value
959D 20 189F            jsr     pntx            ;get address contents in A
95A0 20 C293            jsr     prpntxa         ;print address and data
```

```
95A3 20 32C0            jsr     spa             ;followed by a space
95A6 20 20C0    mond    jsr     input           ;get character from input device
95A9 20 41C0            jsr     loupch          ;convert lower to upper case
95AC C9 27              cmp     #$27            ;if not ASCII display
95AE D0 0F              bne     mond1           ;go test others
95B0 20 23C0            jsr     output          ;else print a '
95B3 20 189F            jsr     pntx            ;get data again
95B6 20 B694            jsr     prgrnr          ;print in ASCII representation
95B9 20 32C0            jsr     spa             ;print a space
95BC 4C A695            jmp     mond            ;and get input
95BF C9 5C      mond1   cmp     #$5c            ;if input was a \
95C1 D0 11              bne     mone
95C3 20 23C0            jsr     output          ;echo it
95C6 20 20C0            jsr     input           ;get input
95C9 C9 20              cmp     #$20            ;if control character
95CB 90 D9              bcc     mond            ;ignore and get input
95CD 8D 269F            sta     inl             ;else save character
95D0 48                 pha                     ;on stack
95D1 4C 0596            jmp     monha           ;echo character, and go get input
95D4 C9 0D      mone    cmp     #$0D            ;if input was RETURN
95D6 D0 06              bne     monf
95D8 20 0E96            jsr     store           ;store data if it was changed
95DB 4C 2FC0            jmp     crlf            ;and end routine
95DE C9 0A      monf    cmp     #$0A            ;if input was Line Feed
95E0 D0 09              bne     mong
95E2 20 0E96            jsr     store           ;store data if it was changed
95E5 20 A093            jsr     pntxinc         ;get next address
95E8 4C 9795            jmp     monca           ;and do next location
95EB C9 0B      mong    cmp     #$0B            ;if character was VT
95ED D0 09              bne     monh
95EF 20 0E96            jsr     store           ;store data if changed
95F2 20 AE93            jsr     pointdec        ;get previous address
95F5 4C 9795            jmp     monca           ;and do previous location
95F8 C9 20      monh    cmp     #$20            ;if other character
95FA 90 AA              bcc     mond            ;ignore control characters
95FC 48                 pha                     ;else save character
95FD 20 6693            jsr     hexnum          ;read data into inl/inh
9600 10 03              bpl     monha           ;if OK, echo and loop
9602 68                 pla                     ;else get character again
9603 D0 A1              bne     mond            ;and get new input (branch always)
                ;
9605 68         monha   pla                     ;get input character
9606 20 23C0            jsr     output          ;echo it
9609 CE 2D9F            dec     aapgew          ;and flag data change
960C D0 98              bne     mond            ;then loop (branch always)
                ;
                ;*** Store the changed data ***
                ;
960E AD 2D9F    store   lda     aapgew          ;was data changed?
9611 F0 0C              beq     strnd           ;if so,
9613 20 2294            jsr     d8pntx          ;make a RAM program
9616 AD 269F            lda     inl             ;get data
9619 20 189F            jsr     pntx            ;store it
961C 20 1694            jsr     adpntx          ;restore old RAM program
961F 20 5D93    strnd   jsr     resin           ;clear input area
9622 8C 2D9F            sty     aapgew          ;reset change flag
9625 60                 rts                     ;and exit
                ;
                ;*** Command V: Compare two blocks of memory ***
                ;(integrated with command M by ndv)
9626 20 1F94    cmv     jsr     cdpntx          ;make RAM programs; compare source
9629 20 3094            jsr     adpnty          ;and fetch destination
962C A9 80              lda     #$80            ;set command V and upward move
962E D0 08              bne     cmv0            ;and go do command
                ;
                ;*** Command M: Move a block of memory ***
                ;(command V integrated by ndv)
9630 20 1694    cmm     jsr     adpntx          ;make RAM program for fetch from source
9633 20 3394            jsr     d8pnty          ;and store in destination
9636 A9 00              lda     #$00            ;set command M and upward move
```

```
9638 8D 2D9F    cmv0    sta     cmpsta          ;in flag
963B 20 CE92            jsr     beg3            ;get 3 parameters, check A<B
963E AD 209F            lda     para1           ;if source
9641 CD 249F            cmp     parcl           ;address is
9644 AD 219F            lda     parah           ;lower than
9647 ED 259F            sbc     parch           ;destination
964A B0 3D              bcs     move0           ;go move downwards
                ;
964C AD 249F            lda     parcl           ;else get destination start lo
964F 6D 229F            adc     parbl           ;add source end lo (carry was clear)
9652 A8                 tay                     ;save result in Y
9653 AD 259F            lda     parch           ;get destination start hi
9656 6D 239F            adc     parbh           ;add carry and source end lo
9659 48                 pha                     ;save on stack
965A 38                 sec                     ;prepare for subtract
965B 98                 tya                     ;get lo
965C ED 209F            sbc     para1           ;subtract source begin lo
965F 8D 249F            sta     parcl           ;and set new destination (end of dest.block)
9662 68                 pla                     ;get result hi back
9663 ED 219F            sbc     parah           ;subtract source begin hi
9666 8D 259F            sta     parch           ;and complete destination end address
9669 AC 209F            ldy     para1           ;get source begin
966C AE 219F            ldx     parah           ;in XY
966F AD 229F            lda     parbl           ;get source end lo
9672 8D 209F            sta     para1           ;as new source start
9675 AD 239F            lda     parbh           ;and complete
9678 8D 219F            sta     parah           ;source start
967B 8C 229F            sty     parbl           ;then set old source start
967E 8E 239F            stx     parbh           ;as source end
                ;(start moving data at end of blocks with decreasing pointers)
9681 AD 2D9F            lda     cmpsta          ;get flag
9684 09 40              ora     #%01000000      ;set direction bit
9686 8D 2D9F            sta     cmpsta          ;and
9689 20 4493    move0   jsr     copypar         ;copy parameters in RAM programs
968C 2C 2D9F    move    bit     cmpsta          ;if command is V
968F 30 03              bmi     movea           ;skip data fetch
9691 20 189F            jsr     pntx            ;get byte from source
9694 20 1C9F    movea   jsr     pnty            ;move to/get destination
9697 2C 2D9F            bit     cmpsta          ;if command is M
969A 10 36              bpl     moveb           ;skip check
969C 20 189F            jsr     pntx            ;else compare with source
969F F0 31              beq     moveb           ;if equal, go test for end address
96A1 48                 pha                     ;save the destination byte
96A2 20 3BC0            jsr     prtext          ;print message
96A5 0D536F7572         fcc     '\rSource: ',0
96AA 63653A2000
96AF 20 1694            jsr     adpntx          ;make RAM program
96B2 20 189F            jsr     pntx            ;get source byte
96B5 20 C293            jsr     prpntxa         ;and print address and data
96B8 20 3BC0            jsr     prtext          ;print message
96BB 2020446573         fcc     '  Dest.: ',0
96C0 742E3A2000
96C5 AE 1E9F            ldx     pntyh           ;get destination address
96C8 AC 1D9F            ldy     pntyl           ;in XY
96CB 68                 pla                     ;get destination data
96CC 20 C893            jsr     pradda          ;print address and data
96CF 20 1F94            jsr     cdpntx          ;change RAM program; compare source
96D2 20 3393    moveb   jsr     checkend        ;if end address reached
96D5 90 17              bcc     movec           ;go end command
96D7 2C 2D9F            bit     cmpsta          ;get direction
96DA 70 09              bvs     downwrd         ;if downwards, go decrement
96DC 20 A093            jsr     pntxinc         ;else increment source
96DF 20 A393            jsr     pntyinc         ;and destination pointers
96E2 4C 8C96            jmp     move            ;and loop
96E5 20 AE93    downwrd jsr     pointdec        ;if downward move,
96E8 20 B193            jsr     pntydec         ;decrement pointers
96EB 4C 8C96            jmp     move            ;and loop (branch always)
96EE 4C 2FC0    movec   jmp     crlf            ;end routine
                ;
                ;********** MON5 **********
```

```
                        ;
                        ;*** Command C: check memory for a certain value ***
                        ;(integrated in F command by ndv)
96F1 20 1F94    cmc     jsr     cdpntx          ;make a RAM program
96F4 A9 FF              lda     #$ff            ;get verify flag
96F6 D0 05              bne     cmc0            ;and go check
                        ;
                        ;*** Command F: Fill RAM ***
                        ;(C command integrated and shortened by ndv)
96F8 20 2294    cmf     jsr     d8pntx          ;make a RAM program
96FB A9 00              lda     #0              ;get store flag
96FD 8D 2D9F    cmc0    sta     cmpsta          ;set it
9700 20 D992            jsr     twpaby          ;get two parameters
9703 20 4493            jsr     copypar         ;copy parameters into RAM program
9706 20 9092            jsr     gchb            ;get next character
9709 C9 27              cmp     #$27            ;if fill value is in ASCII
970B F0 36              beq     fillb           ;go get ASCII
970D CE 169F            dec     bufpoint        ;else adapt buffer pointer
9710 20 FE92            jsr     in              ;and get fill value in hex
9713 20 1F93    fillf   jsr     schtt           ;check for delimiter
9716 D0 37              bne     fillerr         ;if no delimiter, give error
9718 AD 269F    filla   lda     inl             ;get the byte
971B 20 189F            jsr     pntx            ;store/compare the byte
971E 08                 php                     ;save the flags
971F 2C 2D9F    cmcc    bit     cmpsta          ;get C/F flag
9722 10 13              bpl     cmcd            ;if F command, go test for end
9724 28                 plp                     ;else get the flags back
9725 F0 0F              beq     cmcg            ;if equal, go check if at end
9727 20 1694            jsr     adpntx          ;else difference found, change RAM program
972A 20 2FC0            jsr     crlf            ;start on new line
972D 20 189F            jsr     pntx            ;get memory contents
9730 20 C293            jsr     prpntxa         ;print address and data
9733 20 1F94            jsr     cdpntx          ;restore RAM program
9736 24         cmcg    fcc     $24             ;skip next instruction
9737 28         cmcd    plp
9738 20 3393            jsr     checkend        ;if at end address
973B 90 B1              bcc     movec           ;go end command
973D 20 A093            jsr     pntxinc         ;else increment address
9740 4C 1897            jmp     filla           ;and loop
9743 20 9092    fillb   jsr     gchb            ;get fill byte as ASCII
9746 8D 269F            sta     inl             ;save it
9749 20 9092            jsr     gchb            ;get next character
974C 4C 1397            jmp     fillf           ;and go check for delimiter
974F 4C 2092    fillerr jmp     illpar          ;error exit: Illegal parameter
                        ;
                        ;*** Command <: Repeat the inputbuffer ***
                        ;
9752 A9 00      cm.     lda     #$00            ;reset
9754 8D 169F            sta     bufpoint        ;buffer pointer
9757 68                 pla                     ;skip
9758 68                 pla                     ;return address
9759 4C 1391            jmp     main            ;and enter mainloop
                        ;
                        ;*** Command L: List CPU registers ***
                        ;(shortened and disassembly of current pc added by ndv)
975C 20 EA93    cml     jsr     cmdonly         ;check for no parameters
975F 20 3BC0    cmla    jsr     prtext          ;print text
9762 1B6E               fcc     $1b,'n'         ;disable inverse
9764 1B47               fcc     $1b,'G'         ;disable graphics
9766 0D41202058         fcc     '\rA  X  Y  SP PC   NV BDIZC\r',0
976B 2020592020
9770 5350205043
9775 2020204E56
977A 204244495A
977F 430D00
9782 A0 03              ldy     #3              ;get counter
9784 B9 2E9F    1       lda     spuser,y        ;get saved 8-bit register
9787 20 9A93            jsr     prbyts          ;print in hex, followed by a space
978A 88                 dey                     ;point to next register
978B 10 F7              bpl     1.b             ;and loop until all done
```

```
978D AE 339F          ldx    pch          ;get program counter hi
9790 AC 329F          ldy    pcl          ;and lo
9793 8E 1A9F          stx    pntxh        ;set address
9796 8C 199F          sty    pntxl        ;for disassembler
9799 20 D693          jsr    pradd        ;print PC plus a space
979C AD 349F          lda    preg         ;get processor status
979F A2 08    showpra ldx    #8           ;get bitcounter
97A1 0A       spra    asla                ;get bit 7 in carry
97A2 48               pha                 ;save current accu
97A3 20 CB97          jsr    pr01         ;print state of carry as 0 or 1
97A6 68               pla                 ;restore A
97A7 CA               dex                 ;if not 8 bits done
97A8 D0 F7            bne    spra         ;loop
97AA 2C 559F          bit    bkptf1       ;if no breakpoint set
97AD 10 16            bpl    notbkpt      ;go disassemble
97AF AE 1A9F          ldx    pntxh        ;else get PC
97B2 AC 199F          ldy    pntxl        ;in XY
97B5 EC 579F          cpx    bkpth        ;test if
97B8 D0 0B            bne    notbkpt      ;at
97BA CC 569F          cpy    bkptl        ;breakpoint
97BD D0 06            bne    notbkpt      ;if so,
97BF 20 B098          jsr    prbrpt       ;print message
97C2 20 9698          jsr    rsbkpt       ;reset breakpoint
97C5 20 F99B  notbkpt jsr    dis          ;disassemble instruction
97C8 4C 2FC0          jmp    crlf         ;then quit command
                    ;(shortened by ndv)
97CB A9 00    pr01    lda    #0           ;clear result area
97CD 69 30            adc    #'0'         ;convert to ASCII digit
97CF 4C 23C0          jmp    output       ;and print that
                    ;
                    ;*** Command E: execute user program ***
                    ;
97D2 20 9092  cme     jsr    gchb         ;get next character
97D5 C9 20            cmp    #$20         ;if it is a space
97D7 F0 16            beq    cmec         ;go read address
97D9 C9 0D            cmp    #$0d         ;if RETURN
97DB F0 03            beq    cmea         ;go run from current PC
97DD 4C 0092  cmeb    jmp    illcom       ;if other, report error
97E0 AD 329F  cmea    lda    pcl          ;get current PC
97E3 8D 209F          sta    paral        ;in
97E6 AD 339F          lda    pch          ;parameter A
97E9 8D 219F          sta    parah
97EC 4C FA97          jmp    runa         ;and go execute from that address
97EF 20 5D93  cmec    jsr    resin        ;clear parameter A
97F2 20 BE92  run     jsr    eenpar       ;get address in parameter A
97F5 10 03            bpl    runa         ;if no error, go execute
97F7 4C 2092          jmp    illpar       ;else report illegal parameter
97FA AD 219F  runa    lda    parah        ;get startaddress
97FD 48               pha                 ;on
97FE AD 209F          lda    paral        ;stack
9801 48               pha                 ;
9802 AD 349F          lda    preg         ;then restore the flags
9805 48               pha                 ;and
9806 AE 309F          ldx    xreg         ;X
9809 AC 2F9F          ldy    yreg         ;Y
980C AD 319F          lda    acc          ;and accu
980F 40               rti                 ;then execute program pointed to by PARA
                    ;
                    ;*** Command B: Place breakpoint ***
                    ;
9810 20 0194  cmb     jsr    testsp       ;test for a space
9813 20 9092          jsr    gchb         ;get next char.
9816 C9 52            cmp    #'R'         ;if not reset breakpoint
9818 D0 10            bne    cmba         ;go test for others
981A 20 9092          jsr    gchb         ;else get next character
981D C9 0D            cmp    #$0d         ;if not RETURN
981F F0 03            beq    cmbaa
9821 4C 0092  cmbab   jmp    illcom       ;report error
9824 20 9698  cmbaa   jsr    rsbkpt       ;else reset breakpoint
9827 4C 2FC0          jmp    crlf         ;and exit
```

```
                        ;
982A C9 53      cmba    cmp     #'S'            ;test if show breakpoint
982C D0 35              bne     cmbd            ;if not, go get breakpoint address
982E 20 9092            jsr     gchb            ;else get next character
9831 C9 0D              cmp     ##0d            ;if not a RETURN
9833 D0 EC              bne     cmbab           ;report an error
9835 2C 559F            bit     bkptfl          ;else check if there is a breakpoint
9838 30 10              bmi     cmbac           ;if not
983A 20 3BC0            jsr     prtext          ;display message
983D 070D4E6F20         fcc     7,'\rNo b',0
9842 6200
9844 20 B698            jsr     prbrpta         ;print 'reakpoint'
9847 4C 2FC0            jmp     crlf            ;and exit
984A 20 B098    cmbac   jsr     prbrpt          ;print message
984D 20 3BC0            jsr     prtext          ;complete it
9850 3A202400           fcc     ': $',0
9854 AE 579F            ldx     bkpth           ;get address
9857 AC 569F            ldy     bkptl           ;in XY
985A AD 589F            lda     bkptvr          ;get original opcode
985D 20 C893            jsr     pradda          ;print them in hex
9860 4C 2FC0            jmp     crlf            ;and exit
                        ;
9863 CE 169F    cmbd    dec     bufpoint        ;point to previous character
9866 20 BE92            jsr     eenpar          ;get parameter
9869 10 03              bpl     cmbc            ;if error
986B 4C 2092            jmp     illpar          ;show illegal parameter
986E 2C 559F    cmbc    bit     bkptfl          ;if a breakpoint set already
9871 10 03              bpl     cmbb            ;first
9873 20 9698            jsr     rsbkpt          ;reset last breakpoint
9876 20 1694    cmbb    jsr     adpntx          ;make a RAM program
9879 20 4493            jsr     copypar         ;copy parameters into RAM program
987C 8D 569F            sta     bkptl           ;set breakpoint
987F 8E 579F            stx     bkpth           ;address
9882 20 189F            jsr     pntx            ;get opcode from breakpoint address
9885 8D 589F            sta     bkptvr          ;save as well
9888 20 2294            jsr     d8pntx          ;change RAM program
988B A9 00              lda     ##$00           ;get opcode for BRK
988D 20 189F            jsr     pntx            ;place breakpoint
9890 CE 559F            dec     bkptfl          ;flag breakpoint active
9893 4C 2FC0            jmp     crlf            ;then exit
                        ;
9896 20 2294    rsbkpt  jsr     d8pntx          ;make a RAM program
9899 AD 569F            lda     bkptl           ;get breakpoint address
989C 8D 199F            sta     pntxl           ;copy it into RAM program
989F AD 579F            lda     bkpth
98A2 8D 1A9F            sta     pntxh
98A5 A9 00              lda     ##$00           ;reset flag
98A7 8D 559F            sta     bkptfl          ;for breakpoint active
98AA AD 589F            lda     bkptvr          ;get original opcode
98AD 4C 189F            jmp     pntx            ;and restore it, then exit
                        ;
98B0 20 3BC0    prbrpt  jsr     prtext          ;print header
98B3 0D4200             fcc     '\rB',0
98B6 20 3BC0    prbrpta jsr     prtext          ;show breakpoint message
98B9 7265616B70         fcc     'reakpoint',0
98BE 6F696E7400
98C3 60                 rts
                        ;
                        ;*** Command S: calculate a checksum ***
                        ;(shortened by ndv)
98C4 20 F393    cms     jsr     begin           ;get addresses
98C7 20 1694            jsr     adpntx          ;make a RAM program
98CA 20 5D93            jsr     resin           ;clear INL and INH
98CD 18         cmsa    clc                     ;prepare for add
98CE 20 189F            jsr     pntx            ;get byte from memory
98D1 6D 269F            adc     inl             ;add to INL
98D4 8D 269F            sta     inl             ;save result
98D7 90 03              bcc     1.f             ;if carry
98D9 EE 279F            inc     inh             ;adapt hi
98DC 20 3393    1       jsr     checkend        ;test if at end address
```

```
98DF 90 14              bcc     prhlb           ;if so, end command
98E1 20 A093            jsr     pntxinc         ;else increment pointer
98E4 4C CD98            jmp     cmsa            ;and loop
                ;
                ;*** Command $: convert decimal to hex ***
                ;
98E7 20 0194    cm24    jsr     testsp          ;test if followed by a space
98EA 20 A799            jsr     dechex          ;read decimal number into INL and INH
98ED B0 1A              bcs     ilparm          ;if carry set, illegal parameter
98EF F0 0A              beq     overfl          ;if overflow, go report it
98F1 C9 20              cmp     #$20            ;if space only
98F3 F0 14              beq     ilparm          ;report illegal parameter
98F5 20 DE93    prhlb   jsr     prhlb2          ;print result in hex on a new line
98F8 4C 2FC0            jmp     crlf            ;then exit
98FB 20 3BC0    overfl  jsr     prtext          ;overflow found, print message
98FE 20204F7665         fcc     '  Overflow',0
9903 72666C6F77
9908 00
9909 4C 2092    ilparm  jmp     illpar          ;and report error
                ;
                ;*** Command #: convert hex to decimal ***
                ;
990C 20 0194    cm23    jsr     testsp          ;test for a space
990F 20 7199            jsr     gthex           ;read hex input into INL and INH
9912 B0 F5              bcs     ilparm          ;if illegal delimiter, exit with error
9914 F0 E5              beq     overfl          ;if overflow, report that
9916 CE 169F            dec     bufpoint        ;else point to last character
9919 20 9092            jsr     gchb            ;get it
991C C9 20              cmp     #$20            ;if it is a space
991E F0 E9              beq     ilparm          ;exit with error
9920 AD 279F            lda     inh             ;else get hi byte
9923 AE 269F            ldx     inl             ;and lo
9926 20 0F9A            jsr     conver          ;convert hex to decimal
9929 20 2FC0            jsr     crlf            ;start on new line
992C 20 4E9A            jsr     pridec          ;print the decimal number
992F 4C 2FC0            jmp     crlf            ;and exit
                ;
                ;*** Command +: add two hex numbers ***
                ;(carry display added by ndv)
9932 20 0B94    cm2b    jsr     bgncm2          ;get two parameters
9935 18                 clc                     ;prepare for add
9936 AD 209F            lda     para1           ;get 1st lo
9939 6D 229F            adc     parb1           ;add 2nd lo
993C 8D 269F            sta     inl             ;save result
993F AD 219F            lda     parah           ;get 1st hi
9942 6D 239F            adc     parbh           ;add carry and 2nd hi
9945 8D 279F    plusend sta     inh             ;save result hi
9948 08                 php                     ;save the flags
9949 20 DE93            jsr     prhlb2          ;print result on new line
994C 20 3BC0            jsr     prtext          ;print message
994F 2020433D00         fcc     '  C=',0
9954 28                 plp                     ;regain the flags
9955 20 CB97            jsr     pr01            ;print carry as digit
9958 4C 2FC0            jmp     crlf            ;and exit
                ;
                ;*** Command -: subtract two hex numbers ***
                ;
995B 20 0B94    cm2d    jsr     bgncm2          ;get two hex numbers
995E 38                 sec                     ;prepare for subtract
995F AD 209F            lda     para1           ;get 1st lo
9962 ED 229F            sbc     parb1           ;subtract 2nd lo
9965 8D 269F            sta     inl             ;save result
9968 AD 219F            lda     parah           ;get 1st hi
996B ED 239F            sbc     parbh           ;subtract carry and 2nd hi
996E 4C 4599            jmp     plusend         ;then finish command
                ;
                ;********** MON6 **********
                ;
                ;*** ASCII to hex conversion ***
                ;on exit: C=1 is illegal parameter, Z=1 is overflow
```

```
9971 AE 169F    gthex   ldx     bufpoint        ;get input buffer pointer
9974 E8                 inx                     ;point to begin of number
9975 8E 299F            stx     res             ;save start of number
9978 20 FE92            jsr     in              ;get a hex number from inputbuffer
997B C9 0D              cmp     #$0D            ;if RETURN
997D F0 09              beq     lshxa           ;go process number
997F CD 109F            cmp     delim           ;if default delimiter
9982 F0 04              beq     lshxa           ;go process number
9984 C9 3A              cmp     #':'            ;if not colon
9986 D0 10              bne     lshxb           ;exit with error
9988 38         lshxa   sec                     ;prepare for subtract
9989 AD 169F            lda     bufpoint        ;get current buffer pointer
998C ED 299F            sbc     res             ;subtract begin position
998F F0 07              beq     lshxb           ;if zero digits, exit with error
9991 C9 05              cmp     #5              ;if more than four digits
9993 B0 05              bcs     lshxc           ;error
9995 A9 FF              lda     #$ff            ;else set flag for
9997 60                 rts                     ;correct exit
9998 38         lshxb   sec                     ;error exit (illegal parameter)
9999 60                 rts
999A A9 00      lshxc   lda     #$00            ;error exit (overflow)
999C 60                 rts
                ;
                ;*** Convert ASCII byte to decimal nibble ***
                ;(shortened by ndv)
999D 38         ckget   sec                     ;prepare for subtract
999E E9 30              sbc     #'0'            ;if smaller than zero
99A0 90 03              bcc     ckgt1           ;exit with error
99A2 C9 0A              cmp     #10             ;if bigger than nine
99A4 60                 rts                     ;else exit with result
99A5 38         ckgt1   sec                     ;Error exit (illegal parameter)
99A6 60                 rts
                ;
                ;*** Convert from decimal to hex ***
                ;on exit: C=1 is illegal parameter, Z=1 is error
                ;X and Y registers exchanged (ndv)
99A7 A2 00      dechex  ldx     #0              ;X is digit counter
99A9 20 9092    dbt1    jsr     gchb            ;get character
99AC C9 0D              cmp     #$0D            ;if RETURN
99AE F0 1A              beq     dbt3            ;go end
99B0 CD 109F            cmp     delim           ;else must be other delimiter
99B3 F0 15              beq     dbt3            ;if so, end as well
99B5 C9 3A              cmp     #':'            ;colon is
99B7 F0 11              beq     dbt3            ;also valid
99B9 20 9D99            jsr     ckget           ;else convert ASCII byte to decimal nibble
99BC B0 07              bcs     stcker          ;if error, go restore the stack
99BE 48                 pha                     ;else save the nibble on stack
99BF E8                 inx                     ;adapt digit counter
99C0 E0 06              cpx     #6              ;if not 6 digits yet, loop
99C2 D0 E5              bne     dbt1            ;Branch if no error
99C4 18                 clc                     ;else clear parameter error flag
99C5 68         stcker  pla                     ;restore the stack
99C6 CA                 dex                     ;in X number of digits on stack
99C7 D0 FC              bne     stcker          ;until stack pointer correct
99C9 60         dbt2    rts                     ;then exit with error
99CA 8A         dbt3    txa                     ;valid delimiter found
99CB F0 FC              beq     dbt2            ;if zero digits, exit with error
99CD 18                 clc                     ;else prepare for add
99CE 8E 299F            stx     res             ;save number of digits
99D1 20 5D93            jsr     resin           ;clear INL and INH
99D4 88                 dey                     ;set flag (Y=$FF)
99D5 8C 2A9F            sty     eindvlag        ;for correct exit
99D8 A2 00              ldx     #0              ;get initial index
99DA 68         sdh1    pla                     ;get nibble from stack
99DB A8                 tay                     ;decimal nibble is loopcounter now
99DC F0 15              beq     sdh3            ;do nothing if digit is zero
99DE BD 059A    sdh2    lda     dhtba,x         ;else get low byte value from table
99E1 6D 269F            adc     inl             ;add to result
99E4 8D 269F            sta     inl             ;in INL and INH
99E7 BD 0A9A            lda     dhtbb,x         ;get highbyte value from table
```

```
99EA 6D 279F          adc     inh             ;add carry
99ED 8D 279F          sta     inh             ;and complete result
99F0 88               dey                     ;adapt count
99F1 D0 EB            bne     sdh2            ;loop until count is 0
99F3 E8       sdh3    inx                     ;do next digit
99F4 90 05            bcc     sdh4            ;if carry from last digit (result >$FFFF)
99F6 A9 00            lda     #0              ;set flag
99F8 8D 2A9F          sta     eindvlag        ;to error
99FB EC 299F  sdh4    cpx     res             ;all digits done?
99FE D0 DA            bne     sdh1            ;if not, loop
9A00 AD 2A9F  sdh5    lda     eindvlag        ;else get error flag (0=error, $FF=OK)
9A03 18               clc                     ;clear carry
9A04 60               rts                     ;and exit
                      ;
                      ;Table for decimal to hex conversion
                      ;
9A05 01       dhtba   fcc     $01             ;*1
9A06 0A               fcc     $0A             ;*10
9A07 64               fcc     $64             ;*100
9A08 E8               fcc     $E8             ;*1000
9A09 10               fcc     $10             ;*10000
9A0A 00       dhtbb   fcc     $00             ;*1
9A0B 00               fcc     $00             ;*10
9A0C 00               fcc     $00             ;*100
9A0D 03               fcc     $03             ;*1000
9A0E 27               fcc     $27             ;*10000
                      ;
                      ;*** Convert from 2 hex bytes to 3 decimal bytes ***
                      ;
9A0F 8E 2B9F  conver  stx     deci1           ;save lo
9A12 8D 2C9F          sta     deci2           ;and hi
9A15 A9 00            lda     #0              ;and clear
9A17 8D 2D9F          sta     deci3           ;the rest
9A1A AA               tax                     ;clear units
9A1B F8               sed                     ;go into decimal mode
9A1C F0 1B            beq     hdf             ;and start conversion (branch always)
9A1E CE 2C9F  hda     dec     deci2           ;adapt count
9A21 8A               txa                     ;get units
9A22 69 56            adc     #$56            ;Add 56 (decimal)
9A24 AA               tax                     ;save units in X
9A25 68               pla                     ;get hundreds
9A26 69 02            adc     #$02            ;add two
9A28 90 0F            bcc     hdf             ;if carry
9A2A EE 2D9F          inc     deci3           ;adapt tenthousands
9A2D D0 0A            bne     hdf             ;and branch always
9A2F CE 2B9F  hdb     dec     deci1           ;decrement lo byte
9A32 8A               txa                     ;get units
9A33 69 01            adc     #1              ;add one
9A35 AA               tax                     ;save units
9A36 68               pla                     ;get hundreds
9A37 69 00            adc     #0              ;add carry
9A39 48       hdf     pha                     ;save hundreds
9A3A 18               clc                     ;prepare for add
9A3B AD 2B9F          lda     deci1           ;if lo not zero yet
9A3E D0 EF            bne     hdb             ;go calculate lo in decimal in X and stack
9A40 AD 2C9F          lda     deci2           ;if hi not zero yet
9A43 D0 D9            bne     hda             ;go calculate hi in decimal
9A45 D8               cld                     ;else clear decimal mode
9A46 68               pla                     ;get hundreds
9A47 8D 2C9F          sta     deci2           ;into place
9A4A 8E 2B9F          stx     deci1           ;and units as well
9A4D 60               rts                     ;then exit
                      ;
                      ;*** Print decimal numbers ***
                      ;
9A4E AD 2D9F  pridec  lda     deci3           ;get tenthousands
9A51 D0 0B            bne     conoe           ;print no leading zeroes
9A53 AD 2C9F          lda     deci2           ;get hundreds
9A56 D0 15            bne     conof           ;print no leading zeroes
9A58 AD 2B9F          lda     deci1           ;else get units
```

```
9A5B 4C 35C0          jmp   hnout          ;and print least significant byte
9A5E 20 35C0 conoe    jsr   hnout          ;print byte 3 without leading zero
9A61 AD 2C9F          lda   deci2          ;get byte 2
9A64 20 38C0          jsr   prbyte         ;print it
9A67 AD 2B9F conog    lda   deci1          ;get byte 1
9A6A 4C 38C0          jmp   prbyte         ;print it and exit
9A6D 20 35C0 conof    jsr   hnout          ;print byte 2 without leading zero
9A70 4C 679A          jmp   conog          ;and go print byte 1
                      ;
                      ;*** Command W: search for a string ***
                      ;(completely rewritten and replace option added by ndv)
9A73 20 D992 cmw      jsr   twpaby         ;read begin and end address
9A76 20 1994          jsr   bdpntx         ;make a RAM program
9A79 20 4493          jsr   copypar        ;copy parameters into RAMprog.
9A7C A0 00            ldy   #0             ;clear search length counter
9A7E 8C 2A9F          sty   wslength       ;for search
9A81 8C 299F          sty   wrlength       ;and replace values
9A84 20 9092          jsr   gchb           ;get character
9A87 C9 27            cmp   #$27           ;if hex input,
9A89 D0 23            bne   hexin          ;go do hex
9A8B 20 F99A          jsr   rddelm         ;get delimiter
9A8E 20 9692 acsslp   jsr   gchb1          ;get next character
9A91 CD 2B9F          cmp   delimw         ;if it is the delimiter
9A94 F0 08            beq   asrlp          ;go read replace value
9A96 20 059B          jsr   schttw         ;test for CR/:
9A99 20 659B          jsr   stsrch         ;if none, move to buffers
9A9C D0 F0            bne   acsslp         ;and get next character (branch always)
9A9E 20 9692 asrlp    jsr   gchb1          ;get replace character
9AA1 CD 2B9F          cmp   delimw         ;if delimeter
9AA4 F0 42            beq   crtest         ;go test for CR/:
9AA6 20 059B          jsr   schttw         ;else if CR/:, go replace
9AA9 20 729B          jsr   strepl         ;else move to replace buffer
9AAC D0 F0            bne   asrlp          ;and loop
                      ;
9AAE 20 FC9A hexin    jsr   stdelm         ;save delimeter
9AB1 20 FE92 hexslp   jsr   in             ;else read byte
9AB4 AA               tax                  ;move separator to X
9AB5 AD 269F          lda   inl            ;get value
9AB8 20 659B          jsr   stsrch         ;store in buffers
9ABB EC 2B9F          cpx   delimw         ;if separator is the delimiter
9ABE F0 06            beq   hexrin         ;go read replace value
9AC0 8A               txa                  ;get separator in A
9AC1 20 059B          jsr   schttw         ;if CR/:, go search
9AC4 D0 EB            bne   hexslp         ;else get next character
9AC6 20 5D93 hexrin   jsr   resin          ;clear inl/inh
9AC9 20 9092          jsr   gchb           ;get next character
9ACC 20 059B          jsr   schttw         ;if CR/:, go search
9ACF 20 0493          jsr   inb            ;else convert to hex
9AD2 4C D89A          jmp   1.f            ;and go input rest
9AD5 20 FE92 hexrlp   jsr   in             ;get replace byte
9AD8 AA      1        tax                  ;move separator to X
9AD9 AD 269F          lda   inl            ;get value
9ADC 20 729B          jsr   strepl         ;move byte to replace buffer
9ADF 8A               txa                  ;get separator in A
9AE0 20 059B          jsr   schttw         ;if CR/:, go replace
9AE3 CD 2B9F          cmp   delimw         ;if separator is not the delimiter
9AE6 D0 ED            bne   hexrlp         ;go loop, else:
                      ;
9AE8 20 9092 crtest   jsr   gchb           ;get next character
9AEB 20 059B          jsr   schttw         ;if CR or :, go search/replace
9AEE 4C 2092 werr     jmp   illpar         ;else show error
                      ;
9AF1 C0 10   wltest   cpy   #16            ;if max. length reached
9AF3 F0 F9            beq   werr           ;go do error
9AF5 99 459F          sta   wsbuf,y        ;and to replace buffer
9AF8 60               rts                  ;then exit
                      ;
9AF9 20 9692 rddelm   jsr   gchb1          ;get delimiter
9AFC 8D 2B9F stdelm   sta   delimw         ;and save it
9AFF CD 149F          cmp   dntcr          ;if delimeter is the don't care
```

```
9B02 F0 EA              beq     werr            ;show error
9B04 60                 rts                     ;else exit
                        ;
9B05 20 1F93    schttw  jsr     schtt           ;test for CR/:
9B08 F0 01              beq     wexe            ;if found, go execute command
9B0A 60                 rts                     ;else exit
9B0B 68         wexe    pla                     ;remove return address
9B0C 68                 pla                     ;of jsr schttw, and:
9B0D 20 2FC0            jsr     crlf            ;start on new line
9B10 AD 2A9F            lda     wslength        ;get search length
9B13 CD 299F            cmp     wrlength        ;if replace is longer
9B16 90 D6              bcc     werr            ;go show error
9B18 20 3393    cmwf    jsr     checkend        ;if at end address
9B1B 90 54              bcc     5.f             ;exit command, else:
9B1D A2 00              ldx     #0              ;point at start of search value
9B1F EC 2A9F    cmwfaa  cpx     wslength        ;if at end of value
9B22 F0 19              beq     wpradr          ;print address
9B24 20 189F            jsr     pntx            ;else get character from memory
9B27 DD 359F            cmp     whbuf,x         ;compare with buffer
9B2A F0 0E              beq     cmwg            ;if same, compare next
9B2C AD 149F            lda     dntcr           ;if buffer character
9B2F DD 359F            cmp     whbuf,x         ;is don't care
9B32 F0 06              beq     cmwg            ;go test the rest
9B34 20 A093    cmwfa   jsr     pntxinc         ;else increment pointer
9B37 4C 189B            jmp     cmwf            ;and loop
9B3A E8         cmwg    inx                     ;character matches, point to next
9B3B D0 E2              bne     cmwfaa          ;go try next (branch always)
9B3D E0 00      wpradr  cpx     #0              ;if search value is nothing
9B3F F0 F3              beq     cmwfa           ;adapt address
9B41 20 D093            jsr     prpntx          ;else print address
9B44 20 2FC0            jsr     crlf            ;start on new line
9B47 20 2594            jsr     d9pntx          ;change RAM program
9B4A A2 00              ldx     #0              ;start at beginning
9B4C EC 2A9F    2       cpx     wslength        ;if at end,
9B4F F0 0E              beq     4.f             ;go search for next occurrence
9B51 BD 459F            lda     wsbuf,x         ;else get replace byte
9B54 CD 149F            cmp     dntcr           ;if it is the don't care
9B57 F0 03              beq     3.f             ;skip storage
9B59 20 189F            jsr     pntx            ;else move to memory
9B5C E8         3       inx                     ;adapt index
9B5D D0 ED              bne     2.b             ;loop
9B5F 20 1994    4       jsr     bdpntx          ;restore RAM program
9B62 4C 349B            jmp     cmwfa           ;and loop
                        ;
9B65 AC 2A9F    stsrch  ldy     wslength        ;get index
9B68 20 F19A            jsr     wltest          ;test length, move to replace buffer
9B6B 99 359F            sta     whbuf,y         ;move to search buffer
9B6E EE 2A9F            inc     wslength        ;adapt current length
9B71 60         5       rts                     ;exit
                        ;
9B72 AC 299F    strepl  ldy     wrlength        ;get index
9B75 20 F19A            jsr     wltest          ;test length, move to replace buffer
9B78 EE 299F            inc     wrlength        ;adapt replace length
9B7B 60                 rts                     ;and exit
                        ;
                        ;*** Command D: Disassemble a block of memory ***
                        ;
9B7C 20 0194    cmd     jsr     testsp          ;test for space
9B7F F0 03              beq     cmda            ;if not present
9B81 4C 0092            jmp     illcom          ;report illegal command
9B84 20 FE92    cmda    jsr     in              ;else get 1 parameter
9B87 C9 0D              cmp     #$0D            ;if followed by RETURN
9B89 F0 35              beq     cmdpag          ;then disassemble one page
9B8B CD 109F            cmp     delim           ;else if not the correct delimiter
9B8E F0 03              beq     cmdc
9B90 4C 2092    cmdb    jmp     illpar          ;report error
9B93 20 0A93    cmdc    jsr     copya           ;else copy first parameter PARA
9B96 20 1C93            jsr     insch           ;and get next parameter and test for end
9B99 D0 F5              bne     cmdb            ;if wrong end delimiter, give error
9B9B 20 0D93            jsr     copyb           ;else copy second parameter in PARB
```

```
9B9E 20 2693          jsr     chck            ;is PARA<PARB ?
9BA1 B0 03            bcs     cmdd            ;if not,
9BA3 4C 7A92          jmp     zpijlsp         ;show error
9BA6 20 4493  cmdd    jsr     copypar         ;move parameter to RAM program
9BA9 AD 229F  cmde    lda     parbl           ;get end address lo
9BAC CD 199F          cmp     pntxl           ;if current address same, clear C
9BAF AD 239F          lda     parbh           ;get end address hi
9BB2 ED 1A9F          sbc     pntxh           ;clear C if at end or beyond
9BB5 90 06            bcc     cmdf            ;if so, end command
9BB7 20 E99B          jsr     disasm          ;else disassemble one instruction
9BBA 4C A99B          jmp     cmde            ;and loop
9BBD 4C 2FC0  cmdf    jmp     crlf            ;end of command, start of new line
              ;one parameter, so disassemble per page
9BC0 AD 269F  cmdpag  lda     inl             ;copy
9BC3 8D 199F          sta     pntxl           ;parameter
9BC6 AD 279F          lda     inh             ;to
9BC9 8D 1A9F          sta     pntxh           ;zero page pointer
9BCC AD 139F  cmdg    lda     dislct          ;get default line count
9BCF 8D 299F  cmdga   sta     count           ;in counter
9BD2 20 E99B  cmdh    jsr     disasm          ;go disassemble one instruction
9BD5 CE 299F          dec     count           ;adapt line count
9BD8 D0 F8            bne     cmdh            ;if not at end of count, loop
9BDA 20 20C0          jsr     input           ;else get character from input
9BDD C9 0D            cmp     #$0D            ;if RETURN
9BDF F0 DC            beq     cmdf            ;exit command
9BE1 C9 20            cmp     #$20            ;if not space
9BE3 D0 E7            bne     cmdg            ;disassemble another page
9BE5 A9 01            lda     #1              ;else get count of one
9BE7 D0 E6            bne     cmdga           ;and disassemble one line
                      ;
                      ;*** Disassemble one instruction and calculate next opcode address ***
                      ;(shortened by ndv)
9BE9 20 F99B  disasm  jsr     dis             ;disassemble current address
9BEC AD 2A9F          lda     inslen          ;get length of operand(s)
9BEF 20 8E9C          jsr     addoffc         ;calculate next opcode address
9BF2 8C 1A9F          sty     pntxh           ;and store
9BF5 8D 199F          sta     pntxl           ;new instruction pointer
9BF8 60               rts                     ;then exit

                      ;********** MON7 **********
                      ;
                      ;*** Disassembler for 65(C)02, any version ***
                      ;
                      ;       disassembles the instruction pointed to by pntx
                      ;
                      ;       NMOS version originally for Apple ][ by Wozniak/Baum
                      ;       CMOS instructions added by N. de Vries
                      ;       Adapted for MON65 by E.J.M. Visschedijk
                      ;       AS syntax output by N. de Vries
                      ;
9BF9 20 2FC0  dis     jsr     crlf            ;start on new line
9BFC 20 1C94          jsr     b9pntx          ;make a RAM program
9BFF 20 D093          jsr     prpntx          ;print address of instruction
9C02 20 919D          jsr     twspace         ;followed by two spaces
9C05 A0 00            ldy     #0              ;get offset
9C07 20 189F          jsr     pntx            ;get opcode in A
9C0A 20 9D9C          jsr     indexlen        ;determine index and instruction length
9C0D 48               pha                     ;save index
9C0E 20 259D          jsr     prthex          ;print instruction in hex
9C11 68               pla                     ;get index back
9C12 48               pha                     ;save it again
9C13 20 529D          jsr     prtmnem         ;print mnemonic
                      ;
                      ;*** Print the operand(s) of an instruction ***
                      ;
9C16 68               pla                     ;get index back
9C17 10 33            bpl     operpr          ;if positive, print operand
9C19 48               pha                     ;else save index again
9C1A A9 23            lda     #'#'            ;get immediate sign
9C1C 20 23C0          jsr     output          ;print it
```

```
9C1F 20 189F          jsr     pntx            ;get opcode
9C22 4A               lsra                    ;move bit number into place
9C23 4A               lsra
9C24 4A               lsra
9C25 4A               lsra
9C26 29 07            and     #$07            ;get lowest 3 bits
9C28 09 30            ora     #$30            ;convert to ASCII
9C2A 20 23C0          jsr     output          ;print bit number
9C2D A9 2C            lda     #','            ;get a comma
9C2F 20 23C0          jsr     output          ;print it
9C32 68               pla                     ;get index again
9C33 29 02            and     #$02            ;if only one operand
9C35 F0 15            beq     operpr          ;print that
9C37 20 4C9C          jsr     operpr          ;else print first operand
9C3A A9 2C            lda     #','            ;followed by a comma
9C3C 20 23C0          jsr     output
9C3F EE 199F          inc     pntxl           ;adapt 'PC'
9C42 D0 03            bne     npc1            ;if page crossed
9C44 EE 1A9F          inc     pntxh           ;adapt high also
9C47 A9 9E    npc1    lda     #$9E            ;get code for relative
9C49 8D 2D9F          sta     tmp6            ;in addressmode code
9C4C A2 06    operpr  ldx     #6              ;get counter
9C4E E0 03    operlp  cpx     #3              ;if three now
9C50 D0 15            bne     chrtst          ;print address
9C52 AC 2A9F          ldy     inslen          ;else get length
9C55 F0 10            beq     chrtst          ;if zero, print no hex byte
9C57 AD 2D9F  adrlp   lda     tmp6            ;else get mode
9C5A C9 F0            cmp     #$F0            ;check for relative
9C5C 20 189F          jsr     pntx            ;get operand byte
9C5F B0 1D            bcs     reladout        ;if relative, print address and exit
9C61 20 38C0          jsr     prbyte          ;else print high
9C64 88               dey                     ;point to low
9C65 D0 F0            bne     adrlp           ;and print that also
9C67 0E 2D9F  chrtst  asl     tmp6            ;get next bit
9C6A 90 0E            bcc     invchr          ;if set,
9C6C BD 379E          lda     chrtab1-1,x     ;get next character
9C6F 20 23C0          jsr     output          ;print it
9C72 BD 3D9E          lda     chrtab2-1,x     ;get second character
9C75 F0 03            beq     invchr          ;if null, loop
9C77 20 23C0          jsr     output          ;print character
9C7A CA       invchr  dex                     ;adapt counter
9C7B D0 D1            bne     operlp          ;if not all done, loop
9C7D 60               rts                     ;else exit
                      ;
                      ;*** Print relative offset as an absolute address ***
                      ;       (must be entered with carry set)
9C7E 20 8F9C  reladout jsr    addoffs         ;calculate PC+offset+carry
9C81 AA               tax                     ;save the lowbyte in X
9C82 E8               inx                     ;adapt it (now PC+offset+1+carry)
9C83 D0 01            bne     npc2            ;if page crossed
9C85 C8               iny                     ;adapt high
9C86 98       npc2    tya                     ;move highbyte to accu
9C87 20 38C0          jsr     prbyte          ;print it in hex
9C8A 8A               txa                     ;get lowbyte in accu
9C8B 4C 38C0          jmp     prbyte          ;print it in hex and exit
                      ;
                      ;*** Calculate offset+current address in YA ***
                      ;
9C8E 38       addoffc sec                     ;entry to use if carry clear
9C8F AC 1A9F  addoffs ldy     pntxh           ;get highbyte
9C92 AA               tax                     ;set flags according to A
9C93 10 01            bpl     rl1             ;if negative
9C95 88               dey                     ;adapt high
9C96 6D 199F  rl1     adc     pntxl           ;add current low
9C99 90 01            bcc     rl2             ;if page crossed
9C9B C8               iny                     ;adapt high
9C9C 60       rl2     rts                     ;and exit
                      ;
                      ;*** Calculate index, instructionlength and addressmodecode ***
                      ; nctype determines disassembly for NMOS or CMOS
```

```
                ; $40= NMOS, $00= CMOS (only bit 6 is relevant)
                ; c02typ determines disassembly for Rockwell or Standard CMOS
                ; 1= Rockwell, 0=Standard CMOS and NMOS
                ;(both may be changed with command T)
9C9D A8         indexlen tay                    ;save opcode in Y
9C9E 4A                  lsra                   ;get LSB in carry
9C9F 90 0F               bcc     evenop         ;branch if opcode even
9CA1 4A                  lsra                   ;else get bit 1 in carry
9CA2 90 08               bcc     oddop          ;if opcode ends on 11
9CA4 4A                  lsra                   ;get next bit
9CA5 90 23               bcc     illop          ;if clear, illegal instruction
9CA7 AE 119F             ldx     c02typ         ;else get mode
9CAA B0 06               bcs     rockwl         ;get opcode
9CAC 29 07      oddop    and     #$07           ;else opcode ends on 01
9CAE 09 80               ora     #$80           ;add tableoffset
9CB0 4A         evenop   lsra                   ;get next bit in carry
9CB1 AA                  tax                    ;move offset to X
9CB2 2C 129F    rockwl   bit     nctype         ;get NMOS/CMOS flag
9CB5 50 06               bvc     c02type        ;if CMOS, get CMOS table value
9CB7 BD E19D             lda     adrmodn,x      ;else get NMOS
9CBA 4C C09C             jmp     1.f            ;skip CMOS
9CBD BD 9A9D    c02type  lda     adrmod,x       ;get addressmodecode
9CC0 B0 04      1        bcs     bit2on         ;if bit 2 was
9CC2 4A                  lsra                   ;off, move
9CC3 4A                  lsra                   ;high nibble
9CC4 4A                  lsra                   ;to
9CC5 4A                  lsra                   ;low nibble
9CC6 29 0F      bit2on   and     #$0f           ;get low nibble
9CC8 D0 04               bne     validop        ;if zero
9CCA A0 33      illop    ldy     #$33           ;get dummy opcode
9CCC A9 00               lda     #$00           ;and dummy addressmodecode
9CCE AA         validop  tax                    ;get index in lengthtable
9CCF BD 2B9E             lda     lengthtab,x    ;get length and operandcode
9CD2 8D 2D9F             sta     tmp6           ;in TMP6
9CD5 29 03               and     #$03           ;get instructionlength bits
9CD7 F0 02               beq     accm1          ;if accu addressing, do nothing
9CD9 49 03               eor     #%00000011     ;else invert bits
9CDB 8D 2A9F    accm1    sta     inslen         ;save length of operands
9CDE 98                  tya                    ;restore opcode
9CDF C9 9E               cmp     #$9e           ;if STZ ABS,X
9CE1 D0 02               bne     decode
9CE3 A9 9C               lda     #$9c           ;get code for STZ ABS
9CE5 4A         decode   lsra                   ;get LSB in carry
9CE6 90 21               bcc     evenop2        ;if clear do even opcode
9CE8 4A         odd      lsra                   ;else opcode is odd
9CE9 90 0D               bcc     oddop2         ;if opcode ends with 11
9CEB 4A                  lsra                   ;get next bit
9CEC 90 32               bcc     illop2         ;if clear, illegal
9CEE 4A                  lsra                   ;get bit 2 in carry
9CEF 98                  tya                    ;get opcode back
9CF0 2A                  rola                   ;move MSB to carry, C into LSB
9CF1 2A                  rola                   ;convert to
9CF2 29 03               and     #$03           ;opcode index
9CF4 09 E0               ora     #$e0           ;add table offset
9CF6 D0 2A               bne     exit           ;and exit
9CF8 C9 22      oddop2   cmp     #$22           ;if opcode is BIT#
9CFA D0 09               bne     notbit
9CFC 2C 129F             bit     nctype         ;get NMOS/CMOS flag
9CFF 70 C9               bvs     illop          ;if NMOS, point to illegal
9D01 A9 09               lda     #$09           ;else get index to BIT
9D03 D0 1D               bne     exit           ;and exit
9D05 4A         notbit   lsra                   ;else get 3 MSB's
9D06 38                  sec                    ;add tableoffset
9D07 B0 0C               bcs     oddop3         ;and calculate offset
9D09 4A         evenop2  lsra                   ;if opcode ended with 00
9D0A 90 16               bcc     exit           ;exit (index ready now)
9D0C 4A                  lsra                   ;else end was 10
9D0D B0 04               bcs     x6xe           ;if end was 110, do X6XE
9D0F C9 14               cmp     #$14           ;else opcode is LDX #?
9D11 D0 06               bne     x2xa           ;if not, do other X2XA
```

```
9D13 09 20      x6xe    ora     #$20            ;else add table offset
9D15 6A         oddop3  rora                    ;opcode is X6 or XE
9D16 4A         x2      lsra                    ;calculate table offset
9D17 D0 09              bne     exit            ;and exit
9D19 38         x2xa    sec                     ;opcode is X2 or XA
9D1A 6A                 rora                    ;shift in table offset
9D1B 90 F9              bcc     x2              ;if X2, get offset and exit
9D1D 49 D0              eor     #$d0            ;opcode was XA, convert to offset
9D1F 2C                 fcc     $2c             ;skip next instruction
9D20 A9 11      illop2  lda     #$11            ;
9D22 A0 00      exit    ldy     #$00            ;get offset zero (point to opcode)
9D24 60                 rts                     ;and exit
                ;
                ;*** Subroutine PRTHEX ***
                ;
9D25 A2 14      prthex  ldx     #20             ;20 columns object field
9D27 48                 pha                     ;save index
9D28 29 FF              and     #$ff            ;set flags according to [A]
9D2A 10 07              bpl     norock1         ;if positive proceed
9D2C 29 02              and     #$02            ;if XF opcode
9D2E F0 03              beq     norock1         ;
9D30 EE 2A9F            inc     inslen          ;adapt length
9D33 20 189F    norock1 jsr     pntx            ;get instruction byte
9D36 20 38C0            jsr     prbyte          ;print it
9D39 20 32C0            jsr     spa             ;print a space
9D3C CC 2A9F            cpy     inslen          ;set carry if last byte
9D3F CA                 dex
9D40 CA                 dex                     ;minus 1 byte and 1 space
9D41 CA                 dex
9D42 C8                 iny                     ;point to next byte
9D43 90 EE              bcc     norock1         ;if not all done, print that
9D45 68                 pla                     ;get index back
9D46 10 07              bpl     norock2         ;if positive, exit
9D48 29 02              and     #$02            ;if XF code
9D4A F0 03              beq     norock2
9D4C CE 2A9F            dec     inslen          ;get correct instruction length
9D4F 4C 939D    norock2 jmp     xspace          ;and print X spaces
                ;
                ;**** Subroutine PRTMNEM ***
                ;
9D52 29 7F      prtmnem and     #$7f            ;remove MSB from index
9D54 A8                 tay                     ;get real index in Y
9D55 B9 449E            lda     frstlttr,y      ;get encoded first and half second letter
9D58 8D 2B9F            sta     tmp4
9D5B B9 A89E            lda     thrdlttr,y      ;get half second and third letter
9D5E 8D 2C9F            sta     tmp4+$01
9D61 A2 03              ldx     #3              ;get letter counter
9D63 A9 00      lp2     lda     #0              ;clear result area
9D65 A0 05              ldy     #5              ;5 bits to do
9D67 0E 2C9F    lp1     asl     tmp4+$01        ;get
9D6A 2E 2B9F            rol     tmp4            ;next
9D6D 2A                 rola                    ;bit in accu
9D6E 88                 dey                     ;until 5 bits
9D6F D0 F6              bne     lp1             ;done
9D71 69 3F              adc     #'?'            ;then convert to ASCII
9D73 20 23C0            jsr     output          ;print it
9D76 CA                 dex                     ;all letters done?
9D77 D0 EA              bne     lp2             ;if not, repeat
9D79 E8                 inx                     ;get space count
9D7A AD 2D9F            lda     tmp6            ;get addressmode code
9D7D 29 03              and     #%00000011      ;get length indicator bits
9D7F D0 06              bne     noacc           ;if accumulator addressing
9D81 A9 41              lda     #'A'            ;get addressmode designator
9D83 20 23C0            jsr     output          ;print it
9D86 24                 fcc     $24             ;skip next instruction
9D87 E8         noacc   inx
9D88 A9 20              lda     #$20            ;get a space
9D8A 20 23C0    1       jsr     output          ;print it
9D8D CA                 dex                     ;adapt counter
9D8E D0 FA              bne     1.b             ;of not all done, loop
```

```
9D90 60                 rts                     ;else exit
                        ;
                        ;*** Print 2 spaces ***
                        ;
9D91 A2 02      twspace ldx     #2              ;get count, and:
                        ;
                        ;*** Print X spaces ***
                        ;
9D93 20 32C0    xspace  jsr     spa             ;print a space
9D96 CA                 dex                     ;adapt count
9D97 D0 FA              bne     xspace          ;if not all done, loop
9D99 60                 rts                     ;else exit
                        ;
                        ;*** Addressing mode table, 65C02 ***
                        ;
                        ;Each nibble represents an address mode
                        ;
                        ; 0 = Illegal opcode
                        ; 1 = Immediate
                        ; 2 = Zero page
                        ; 3 = Absolute
                        ; 4 = Implied
                        ; 5 = Accumulator
                        ; 6 = Indexed indirect
                        ; 7 = Indirect indexed
                        ; 8 = Zero page
                        ; 9 = Absolute,X
                        ; A = Absolute,Y
                        ; B = Indirect
                        ; C = Zero page,Y
                        ; D = Relative
                        ; E = Absolute Indexed Indirect
                        ; F = Indirect zero page
                        ;
9D9A 40224533DF adrmod  fcc     $40,$22,$45,$33,$df,$28,$45,$39
9D9F 284539
9DA2 30224533DF         fcc     $30,$22,$45,$33,$df,$88,$45,$99
9DA7 884599
9DAA 40024533DF         fcc     $40,$02,$45,$33,$df,$08,$44,$09
9DAF 084409
9DB2 402245B3DF         fcc     $40,$22,$45,$b3,$df,$88,$44,$e9
9DB7 8844E9
9DBA D0224433DF         fcc     $d0,$22,$44,$33,$df,$8c,$44,$39
9DBF 8C4439
9DC2 11224433DF         fcc     $11,$22,$44,$33,$df,$8c,$44,$9a
9DC7 8C449A
9DCA 10224433DF         fcc     $10,$22,$44,$33,$df,$08,$44,$09
9DCF 084409
9DD2 10224433DF         fcc     $10,$22,$44,$33,$df,$08,$44,$09
9DD7 084409
9DDA 621378A900         fcc     $62,$13,$78,$a9,$00,$01,$00
9DDF 0100
                        ;
                        ;*** Addressing mode table, 6502 ***
                        ;
                        ;Each nibble represents an addressmode
                        ;
                        ; 0 = Illegal opcode
                        ; 1 = Immediate
                        ; 2 = Zero page
                        ; 3 = Absolute
                        ; 4 = Implied
                        ; 5 = Accumulator
                        ; 6 = Indexed indirect
                        ; 7 = Indirect indexed
                        ; 8 = Zero page
                        ; 9 = Absolute,X
                        ; A = Absolute,Y
                        ; B = Indirect
                        ; C = Zero page,Y
```

```
                ; D = Relative
                ; E = Not used
                ; F = Not used
                ;
9DE1 40024503D0 adrmodn fcc     $40,$02,$45,$03,$d0,$08,$40,$09
9DE6 084009
9DE9 30224533D0         fcc     $30,$22,$45,$33,$d0,$08,$40,$09
9DEE 084009
9DF1 40024533D0         fcc     $40,$02,$45,$33,$d0,$08,$40,$09
9DF6 084009
9DF9 400245B3D0         fcc     $40,$02,$45,$b3,$d0,$08,$40,$09
9DFE 084009
9E01 00224433D0         fcc     $00,$22,$44,$33,$d0,$8c,$44,$00
9E06 8C4400
9E09 11224433D0         fcc     $11,$22,$44,$33,$d0,$8c,$44,$9a
9E0E 8C449A
9E11 10224433D0         fcc     $10,$22,$44,$33,$d0,$08,$40,$09
9E16 084009
9E19 10224433D0         fcc     $10,$22,$44,$33,$d0,$08,$40,$09
9E1E 084009
9E21 621378A900         fcc     $62,$13,$78,$a9,$00,$01,$00
9E26 0100
                ;
                ;*** Opcode length table ***
                ;
                ;Lowest two bits represent inverted instructionlength minus 1
                ;If both bits are clear, accu addressing is used
                ;
9E28 03         lengthtab fcc   %00000011       ;Illegal opcode
9E29 22                 fcc     %00100010       ;Immediate
9E2A 82                 fcc     %10000010       ;Immediate
9E2B 81                 fcc     %10000001       ;Absolute
9E2C 03                 fcc     %00000011       ;Implied
9E2D 00                 fcc     %00000000       ;Accumulator
9E2E 5A                 fcc     %01011010       ;Indexed indirect [$..,X]
9E2F 4E                 fcc     %01001110       ;Indirect indexed [$..],Y
9E30 92                 fcc     %10010010       ;Zero page,X
9E31 91                 fcc     %10010001       ;Absolute,X
9E32 85                 fcc     %10000101       ;Absolute,Y
9E33 49                 fcc     %01001001       ;Absolute indirect [$....]
9E34 86                 fcc     %10000110       ;Zeropage,Y
9E35 9E                 fcc     %10011110       ;Relative
9E36 59                 fcc     %01011001       ;Absolute indexed indirect [$....,X]
9E37 4A                 fcc     %01001010       ;Zero page indirect [$..]

                ;*** Character table for operand printout ***
                ;
                ;Table is used backwards, starting with offset X=6
                ;
                ;X=6: $     (before hex oper.)
                ;X=5: [$    (before hex oper.)
                ;X=4: #$    (before hex oper.)
                ;X=3: ,X    (after hex oper.)
                ;X=2: ]     (after hex oper.)
                ;X=1: ,Y    (after hex oper.)
                ;X=0:       not used
                ;
                ;
9E38 2C5D2C235B chrtab1 fcc     ',],#[$'        ;First character of a pair
9E3D 24
9E3E 5900582424 chrtab2 fcc     'Y',0,'X$$',0   ;Second character
9E43 00

                ;
                ;*** Mnemonic table ***
                ;
                ;Each mnemonic is represented by two bytes,
                ;which contain three offsets to ASCII.
                ;Each offset is 5 bits long.
                ;When the 5 bit offset is added to $3F, a letter
```

```
                ;or questionmark is obtained.
                ;
9E44            frstlttr
                ;               BRK TSB PHP TSB BPL TRB CLC TRB
9E44 1CAD8AAD1C         fcc     $1c,$ad,$8a,$ad,$1c,$ac,$23,$ac
9E49 AC23AC
                ;               JSR BIT PLP BIT BMI BIT SEC BIT
9E4C 5D1A8B1A1B         fcc     $5d,$1a,$8b,$1a,$1b,$1a,$a1,$1a
9E51 1AA11A
                ;               RTI ??? PHA JMP BVC ??? CLI ???
9E54 9D008A5B1D         fcc     $9d,$00,$8a,$5b,$1d,$00,$23,$00
9E59 002300
                ;               RTS STZ PLA JMP BVS STZ SEI JMP
9E5C 9DA58B5B1D         fcc     $9d,$a5,$8b,$5b,$1d,$a5,$a1,$5b
9E61 A5A15B
                ;               BRA STY DEY STY BCC STY TYA STZ
9E64 1CA529A519         fcc     $1c,$a5,$29,$a5,$19,$a5,$ae,$a5
9E69 A5AEA5
                ;               LDY LDY TAY LDY BCS LDY CLV LDY
9E6C 6969A86919         fcc     $69,$69,$a8,$69,$19,$69,$23,$69
9E71 692369
                ;               CPY CPY INY CPY BNE ??? CLD ???
9E74 242453241B         fcc     $24,$24,$53,$24,$1b,$00,$23,$00
9E79 002300
                ;               CPX CPX INX CPX BEQ ??? SED ???
9E7C 2424532419         fcc     $24,$24,$53,$24,$19,$00,$a1,$00
9E81 00A100
                ;               ORA AND EOR ADC STA LDA CMP SBC
9E84 84133411A5         fcc     $84,$13,$34,$11,$a5,$69,$23,$a0
9E89 6923A0
                ;               ASL ROL LSR ROR STX LDX DEC INC
9E8C 159C6D9CA5         fcc     $15,$9c,$6d,$9c,$a5,$69,$29,$53
9E91 692953
                ;               ASL INC ROL DEC LSR PHY ROR PLY
9E94 15539C296D         fcc     $15,$53,$9c,$29,$6d,$8a,$9c,$8b
9E99 8A9C8B
                ;               TXA TXS TAX TSX DEX PHX NOP PLX
9E9C AEAEA8AD29         fcc     $ae,$ae,$a8,$ad,$29,$8a,$7c,$8b
9EA1 8A7C8B
                ;               RMB SMB BBR BBS
9EA4 9BA31818           fcc     $9b,$a3,$18,$18
9EA8            thrdlttr
                ;               BRK TSB PHP TSB BPL TRB CLC TRB
9EA8 D80662065A         fcc     $d8,$06,$62,$06,$5a,$c6,$48,$c6
9EAD C648C6
                ;               JSR BIT PLP BIT BMI BIT SEC BIT
9EB0 26AA62AA94         fcc     $26,$aa,$62,$aa,$94,$aa,$88,$aa
9EB5 AA88AA
                ;               RTI ??? PHA JMP BVC ??? CLI ???
9EB8 540044A2C8         fcc     $54,$00,$44,$a2,$c8,$00,$54,$00
9EBD 005400
                ;               RTS STZ PLA JMP BVS STZ SEI JMP
9EC0 687644A2E8         fcc     $68,$76,$44,$a2,$e8,$76,$94,$a2
9EC5 7694A2
                ;               BRA STY DEY STY BCC STY TYA STZ
9EC8 C474B47408         fcc     $c4,$74,$b4,$74,$08,$74,$84,$76
9ECD 748476
                ;               LDY LDY TAY LDY BCS LDY CLV LDY
9ED0 7474B4742B         fcc     $74,$74,$b4,$74,$28,$74,$6e,$74
9ED5 746E74
                ;               CPY CPY INY CPY BNE ??? CLD ???
9ED8 7474F474CC         fcc     $74,$74,$f4,$74,$cc,$00,$4a,$00
9EDD 004A00
                ;               CPX CPX INX CPX BEQ ??? SED ???
9EE0 7272F272A4         fcc     $72,$72,$f2,$72,$a4,$00,$8a,$00
9EE5 008A00
                ;               ORA AND EOR ADC STA LDA CMP SBC
9EE8 C4CA264844         fcc     $c4,$ca,$26,$48,$44,$44,$a2,$c8
9EED 44A2C8
                ;               ASL ROL LSR ROR STX LDX DEC INC
```

```
9EF0 1A1A262672         fcc     $1a,$1a,$26,$26,$72,$72,$88,$c8
9EF5 7288C8
                ;               ASL INC ROL DEC LSR PHY ROR PLY
9EF8 1AC81A8826         fcc     $1a,$c8,$1a,$88,$26,$74,$26,$74
9EFD 742674
                ;               TXA TXS TAX TSX DEX PHX NOP PLX
9F00 4468B232B2         fcc     $44,$68,$b2,$32,$b2,$72,$22,$72
9F05 722272
                ;               RMB SMB BBR BBS
9F08 8686E6E8           fcc     $86,$86,$e6,$e8
                ;
                ;********** MON9 **********
                ;
                ;*** Command A: assemble one instruction in AS syntax
                ;(written by ndv)
9F0C 60         cma     rts     ;T.b.s.

         9000           end     mon65
```

global labels

aapgew	9F2D	acc	9F31	accm1	9CDB	acsslp	9A8E	addoffc	9C8E
addoffs	9C8F	adpntx	9416	adpnty	9430	adrlp	9C57	adrmod	9D9A
adrmodn	9DE1	ashett	9382	asrlp	9A9E	b9pntx	941C	bdpntx	9419
beg3	92CE	beg3a	92F2	beg3err	92D6	begin	93F3	beginc	93FE
bgncm2	940B	bif2on	9CC6	bkptf1	9F55	bkpth	9F57	bkpt1	9F56
bkptvr	9F58	break	908C	brkvector	E7B3	brkvs	9FA2	bufloop	912F
bufpoint	9F16	c02typ	9F11	c02type	9CBD	cdpntx	941F	chck	9326
check	9335	checkend	9333	chrtab1	9E38	chrtab2	9E3E	chrtst	9C67
ckget	999D	ckgt1	99A5	cm.	9752	cm23	990C	cm24	98E7
cm2b	9932	cm2d	995B	cma	9F0C	cmaap	9573	cmb	9810
cmba	982A	cmbaa	9824	cmbab	9821	cmbac	984A	cmbb	9876
cmbc	986E	cmbd	9863	cmc	96F1	cmc0	96FD	cmcc	971F
cmcd	9737	cmcg	9736	cmd	9B7C	cmda	9B84	cmdb	9B90
cmdc	9B93	cmdd	9BA6	cmde	9BA9	cmdf	9BBD	cmdg	9BCC
cmdga	9BCF	cmdh	9BD2	cmdonly	93EA	cmdpag	9BC0	cme	97D2
cmea	97E0	cmeb	97DD	cmec	97EF	cmf	96F8	cmh	943E
cml	975C	cmla	975F	cmm	9630	cmoscpu	9536	cmp	94E6
cmpsta	9F2D	cmq	9543	cms	98C4	cmsa	98CD	cmt	94EE
cmv	9626	cmv0	9638	cmw	9A73	cmwf	9B18	cmwfa	9B34
cmwfaa	9B1F	cmwg	9B3A	cmx	91FA	cmy	91FD	commtable	91CC
comtab	91B5	conoe	9A5E	conof	9A6D	conog	9A67	conver	9A0F
cony	9174	copy	930F	copya	930A	copyb	930D	copypar	9344
count	9F29	cr	9134	crlf	C02F	crtest	9AE8	cruit	9138
d8pntx	9422	d8pnty	9433	d9pntx	9425	dbt1	99A9	dbt2	99C9
dbt3	99CA	dechex	99A7	deci1	9F2B	deci2	9F2C	deci3	9F2D
decode	9CE5	delet	914C	delim	9F10	delimw	9F2B	dhtba	9A05
dhtbb	9A0A	dis	9BF9	disasm	9BE9	dislct	9F13	dntcr	9F14
downwrd	96E5	driepar	92A0	eenpar	92BE	eindvlag	9F2A	error1	91A3
evenop	9CB0	evenop2	9D09	exit	9D22	filla	9718	fillb	9743
fillerr	974F	fillf	9713	frstlttr	9E44	gchb	9290	gchb1	9296
gthex	9971	hda	9A1E	hdb	9A2F	hdf	9A39	hexdd	9453
hexdeaq	9468	hexdeq	945F	hexdf	9473	hexdg	948B	hexdh	949B
hexdi	94A5	hexdl	94AB	hexin	9AAE	hexnum	9366	hexrin	9AC6
hexrlp	9AD5	hexslp	9AB1	hfderr	90F7	hnout	C035	hnua	936D
hnub	937F	hoofdprog	90CD	illcm2	925F	illcom	9200	illcom2	925C
illcoma	9203	illop	9CCA	illop2	9D20	illpar	9220	illpara	9223
illparb	9226	ilparm	9909	in	92FE	ina	9301	inb	9304
incpnt	94D7	indexlen	9C9D	inh	9F27	inl	9F26	inpbuffer	9FB0
input	C020	insch	931C	inslen	9F2A	invchr	9C7A	lengthtab	9E28
looktab	9199	loupch	C041	lp1	9D67	lp2	9D63	lshxa	9988
lshxb	9998	lshxc	999A	main	9113	mainpgm	90FB	mon65	9000
monb	9587	monbeg	9000	monc	9594	monca	9597	mond	95A6
mond1	95BF	mone	95D4	monf	95DE	mong	95EB	monh	95F8
monha	9605	move	968C	move0	9689	movea	9694	moveb	96D2
movec	96EE	nctype	9F12	ndvrs	9F59	nmi	90A3	nmivector	E7B1
nmivs	9FA4	nmoscpu	953B	noacc	9D87	nolttro	9408	norock1	9D33
norock2	9D4F	notbit	9D05	notbkpt	97C5	notvat	9392	npc1	9C47
npc2	9C86	odd	9CE8	oddop	9CAC	oddop2	9CF8	oddop3	9D15
operlp	9C4E	operpr	9C4C	output	C023	overf1	98FB	parah	9F21
paral	9F20	parbh	9F23	parbl	9F22	parch	9F25	parcl	9F24
parer	92CB	pch	9F33	pc1	9F32	pijl	926A	plusend	9945
pntdce	93C1	pntx	9F18	pntxh	9F1A	pntxhsave	9F27	pntxinc	93A0
pntxinca	93AD	pntxl	9F19	pntxlsave	9F26	pnty	9F1C	pntydec	93B1
pntyh	9F1E	pntyinc	93A3	pntyl	9F1D	pointdec	93AE	pr01	97CB
pradd	93D6	pradda	93C8	prbffg	9F28	prbrpt	98B0	prbrpta	98B6
prbufpijl	9245	prbyte	C038	prbyts	939A	prcnt	9F59	preg	9F34
presc	94CC	prgrf	94CA	prgrg	94C7	prgrna	94BD	prgrnr	94B6
prhlb	98F5	prhlb2	93DE	pridec	9A4E	prin	93E1	prompt	9F90
prpntx	93D0	prpntxa	93C2	prtext	C03B	prthex	9D25	prtmnem	9D52
qcomvr	9FA1	rddelm	9AF9	reladout	9C7E	release	3630	res	9F29
reserve	9F96	resin	935D	rl1	9C96	rl2	9C9C	rock	9527
rockw1	9CB2	rsbkpt	9896	rsta	904D	rstb	9057	run	97F2
runa	97FA	savstackp	9F15	schtt	931F	schttend	9325	schttw	9B05
sdh1	99DA	sdh2	99DE	sdh3	99F3	sdh4	99FB	sdh5	9A00
setvar	905D	setvr1	905F	setvr2	9070	showpra	979F	showtyp	950B
spa	C032	spra	97A1	spuser	9F2E	stabuf	9157	stcker	99C5
stdelm	9AFC	store	960E	strep1	9B72	strnd	961F	stsrch	9B65
svaint	E770	svstck	9F17	swint	90A0	swivector	E76E	swivs	9FA6

tabvec	9FA8	testsp	9401	testspa	93F2	thrdlttr	9EA8	tmp4	9F2B
tmp6	9F2D	tonmos	94FC	torock	9503	tweepar	92AE	twpaby	92D9
twpend	92FD	twspace	9D91	uitbuffer	9185	userx	9FAA	usery	9FAC
v	9F10	validop	9CCE	valt	9397	valt1	9395	vart1	9073
vart2	9080	ver	2E32	vulbuf	912A	werr	9AEE	wexe	9B0B
whbuf	9F35	wltest	9AF1	wpradr	9B3D	wrlength	9F29	wsbuf	9F45
wslength	9F2A	x2	9D16	x2xa	9D19	x6xe	9D13	xaout	94CE
xreg	9F30	xspace	9D93	yreg	9F2F	zkvsp	92B0	zoek	9283
zpijlsp	927A								

Errors detected: 0