```
;*********************************************
;*                                           *
;* EP                                        *
;*                                           *
;* A driver program for the NdV EPROMGRAMMER. *
;*                                           *
;* Programs and reads the following EPROMs:   *
;* 2516, 2516A, 2716                         *
;* 2532, 2532A, 2732, 2732A                  *
;* 2564, 2564A, 2764, 2764A                  *
;* 27128, 27128A                             *
;* 27256, 27256A                             *
;* 27512                                     *
;*                                           *
;* The following algorithms are available:    *
;* - Classic 50 ms                           *
;* - Inteligent                              *
;* - Quick pulse                             *
;*                                           *
;*********************************************
;
; Name: EP.MAC
;
; Written by:   Nico de Vries
;               Mari Andriessenrade 49
;               2907 MA  Capelle aan den IJssel
;               The Netherlands
;
; History:
;
; 08-03-1987    Start of work
; 28-03-1987    Command input working
; 29-03-1987    Commands D, Q, M and R working
; 07-04-1987    Commands V, E and T working
; 12-04-1987    Command P working; first EPROMs programmed
;               Commands now called as subroutines
; 13-04-1987    Next failure feature added to V command
;               T command tries empty test first
; 15-04-1987    D command modified
;               B command working
; 17-04-1987    C command added
;               Program moved from $A000 to $200
; 30-04-1987    I command added
;               MON65 automatically loaded on start up
;               Editing in B command now possible
;               S command added
; 01-05-1987    V0.99: pre-release for limited distribution
;               HELP-text added and bug fixed
;               LO S:MON replaced by LO S:MONITOR
; 18-05-1987  a bug in I command fixed
;             b use of zero page reduced
;             c code shortened
;             d 2564 intel modes removed
;             e TI A-types added
; 21-05-1987  f bug fix in command entry: LDX $ncmds-1!
; 22-05-1987  g S command switches Vcc off between displays
; 23-06-1987  h Vcc switched off after V-error with No more differences
;             i Break during programming switches supplies off
; 23-06-1987    V1.00: release (identical to V0.99i)
;
; Memory layout:
; 0000-00FF     Zero page
; 0100-01FF     Stack area
; 0200-0FFF     This program
; 1000-8FFF     RAM, used to hold EPROM contents
; 9000-9FFF     Room for MON65
; A000-BFFF     Room for utility
; C000-DFFF     DOS65
; E000-E3FF     I/O area
; E400-E7FF     Scratchpad RAM for IO65 and EXETIME
; E800-EFFF     Video RAM
; F000-FFFF     Boot-ROM
;
; Description of VIA and PIA use:
;
; VIA:
; port A: databus from/to EPROM
; Port B: PB0-PB2  EPROM type select
;         PB3      Switches Vcc from 5 to 6 Volts (pin 28 only)
;         PB4      Vcc on/off switch
;         PB5      Vpp on/off switch
;         PB6      OE drive
;         PB7      Program pulse, generated through timer 1
;         CB2      Vpp A/B select
;
; PIA:
; Port A: addresslines A0 through A7 to EPROM
; Port B: addresslines A8 through A15 to EPROM
;
;*** External Labels
;
3156  rev     equ     "1V"            ;note: backwards!
3030  release equ     "00"            ;note: backwards!
              argl    2               ;default speed is 2 MHz
      arg     argm    cpuclk          ;cpuspeed is -a argument
E7B3  brkvector equ   $e7b3
;
;*********************************************
;*                                           *
;* EP.LIB                                    *
;*                                           *
;* External labels for EP.MAC                *
;*                                           *
;*********************************************
;
; Name: EP.LIB
;
; Written by:   Nico de Vries
;               Mari Andriessenrade 49
;               2907 MA  Capelle aan den IJssel
;               The Netherlands
;
E180  epbase  equ     $e180           ;set to base address of card
E180  via     equ     epbase
E180  vorb    equ     via             ;VIA output register B (control lines)
E181  vora    equ     via+1           ;VIA output register A (data bus to EPROM)
E182  vddrb   equ     via+2           ;VIA data direction register port B
E183  vddra   equ     via+3           ;VIA data direction register port A
E184  vt1cl   equ     via+4           ;VIA timer 1 latch/counter, lo
```

```
                E185  vt1ch   equ     via+5           ;VIA timer 1 counter, hi
                E186  vt1ll   equ     via+6           ;VIA timer 1 latch, lo
                E187  vt1lh   equ     via+7           ;VIA timer 1 latch, hi
                E188  vt2cl   equ     via+8           ;VIA timer 2 latch/counter, lo
                E189  vt2ch   equ     via+9           ;VIA timer 2 counter, hi
                E18A  vsr     equ     via+10          ;VIA shift register
                E18B  vacr    equ     via+11          ;VIA auxiliary control register
                E18C  vpcr    equ     via+12          ;VIA peripheral control register
                E18D  vifr    equ     via+13          ;VIA interrupt flag register
                E18E  vier    equ     via+14          ;VIA interrupt enable register
                E18F  voranh  equ     via+15          ;VIA port A, no handshake
                      ;
                E190  pia     equ     epbase+16       ;base address of PIA
                E190  pora    equ     pia             ;PIA port A/data direction register A
                E191  pcra    equ     pia+1           ;PIA control register A
                E192  porb    equ     pia+2           ;PIA port B/data direction register B
                E193  pcrb    equ     pia+3           ;PIA control register B
                      ;
                      ;*** DOS labels
                      ;
                9000  mon     equ     $9000           ;start address MON65
                9000  ramsize equ     mon             ;upper RAM limit
                C006  doscom  equ     $c006           ;execute dos command pointed to by AY
                C020  input   equ     $c020           ;get input character in A
                C023  output  equ     $c023           ;print the character in A
                C029  bufin   equ     $c029           ;get command into buffer
                C02C  getbuf  equ     $c02c           ;read command from buffer
                C02F  crlf    equ     $c02f           ;output CRLF
                C032  spa     equ     $c032           ;output a space
                C035  hnout   equ     $c035           ;print A as one or two hex nibbles
                C038  prbyte  equ     $c038           ;print A as two hex nibbles
                C03B  prtext  equ     $c03b           ;print string until zero byte
                C041  loupch  equ     $c041           ;convert lower to upper case
                      ;
                      ;*** IO65 labels
                      ;
                E701  stattog equ     $e701           ;inhibit status line update flag
                F01B  statoff equ     $f01b           ;switch status line off
                F021  staton  equ     $f021           ;switch status line on
                F024  crsadr  equ     $f024           ;move cursor to X,Y
                F027  crsback equ     $f027           ;restore previous cursor position
                      ;
                      ;*** Page zero variables
                      ;
                0080          org     $80
                      ;
0080                  rcurr   res     2               ;current RAM address
                      ;
                0200          org     $0200
                      ;
                      ;*** Initialise VIA and PIA, then set all pins
                      ;*** of textool socket at zero volts
                      ;
0200 4C 7202    ep    jmp     ep1             ;skip info part and variables
0203 19         avar  fcb     25              ;number of initial pulses, intel mode
0204 03         bvar  fcb     3               ;overprogram factor, intel mode
0205 C8C5CCD0         fcc     $c8,$c5,$cc,$d0 ;flag for help text
0209 4C4C4C           fcc     $4c,$4c,$4c     ;dummy bytes
020C 4550206373       fcc     'EP is the driver program for the DOS65'
0211 2074686520
0216 6472697665
021B 722070726F
0220 6772616D20
0225 666F722074
022A 686520444F
022F 533635
0232 204550524F       fcc     ' EPROM-programmer.'
0237 4D2D70726F
023C 6772616D6D
0241 65722E
0244 0D456E7465       fcc     '\rEnter EP <CR> and type ? <CR> for more help.'
0249 7220455020
024E 3C43523E20
0253 616E642074
0258 797065203F
025D 203C43523E
0262 20666F7220
0267 6D6F726520
026C 68656C702E
0271 00               fcc     0               ;end of help text
0272 A0 FF      ep1   ldy     #$ff            ;get value for DDR registers
0274 A9 00            lda     #$00            ;and value for DDR access
0276 8D 91E1          sta     pcra            ;set DDRA accessible
0279 8D 93E1          sta     pcrb            ;set DDRB accessible
027C 8C 90E1          sty     pora            ;set port A to outputs
027F 8C 92E1          sty     porb            ;set port B to outputs
0282 A9 04            lda     #%00000100      ;get value for port access
0284 8D 91E1          sta     pcra            ;set port A accessible
0287 8D 93E1          sta     pcrb            ;set port B accessible
028A C8               iny                     ;Y=0
028B 8C 90E1          sty     pora            ;set all addresslines
028E 8C 92E1          sty     porb            ;to zero
0291 98               tya                     ;A=0
0292 88               dey                     ;Y=$FF
0293 8C 83E1          sty     vddra           ;set VIA port A to outputs
0296 8D 8FE1          sta     voranh          ;set port A (databus) to zero volts
0299 8C 82E1          sty     vddrb           ;set VIA port B to outputs
029C 8D 80E1          sta     vorb            ;and set all controllines to zero
029F A9 80            lda     #%10000000      ;set auxiliary control to
02A1 8D 8BE1          sta     vacr            ;Timer1 one shot mode with PB7 pulsing
02A4 8D 8EE1          sta     vier            ;disable all interrupts
02A7 A9 CC            lda     #%11001100      ;set CB2, CA2 to output, lo
02A9 8D 8CE1          sta     vpcr            ;CA1, CB1 negative edge
02AC AD AC0E          lda     epdflt          ;get default type
02AF 8D 2B0F          sta     eptyp           ;into current type
02B2 A9 04            lda     #dfltyp         ;initialize
02B4 8D 2C0F          sta     tabind          ;table index
02B7 20 A00C          jsr     settyp          ;reflect type on VIA
02BA 20 3BC0          jsr     prtext          ;print sign-on message
02BD 0C20455020       fcc     $0c,' EP '
02C2 5631             fdb     rev
02C4 2E               fcc     '.'
02C5 3030             fdb     release
02C7 1B690D2020       fcc     $1b,'i\r  by NdV  ',$1b,'n\r'
02CC 6279204E64
02D1 5620201B6E
02D6 0D
02D7 00               fcc     0
02D8 A9 0F            lda     #monstr>>8      ;point AY
02DA A0 10            ldy     #monstr&255     ;to command string
```

```
02DC 20 06C0            jsr     doscom          ;load MON65
02DF 20 3303            jsr     qtextpr         ;print command list
02E2 20 100C            jsr     showbnd         ;show current type and boundaries
                ;
                ;*** Command entry loop
                ;
02E5 20 2FC0    nextcmd jsr     crlf            ;start on new line
02E8 20 F40C            jsr     prtyp           ;show current EPROM type
02EB 20 3BC0    1       jsr     prtext          ;print prompt
02EE 3E2000             fcc     '> ',0
02F1 20 20C0    2       jsr     input           ;get command character
02F4 C9 0D              cmp     #$0d            ;if return only
02F6 F0 F9              beq     2.b             ;ignore
02F8 20 41C0            jsr     loupch          ;else convert to upper case
02FB A2 0C              ldx     #ncmds-1        ;set command counter
02FD DD 810E    3       cmp     cmds,x          ;check if valid command
0300 F0 05              beq     4.f             ;if so, execute it
0302 CA                 dex                     ;else adapt counter
0303 10 F8              bpl     3.b             ;loop
0305 30 EA              bmi     2.b             ;else get next character
                ;
0307 20 23C0    4       jsr     output          ;and echo input
030A 20 20C0    5       jsr     input           ;get next character
030D C9 0D              cmp     #$0d            ;if return
030F F0 0E              beq     6.f             ;go execute command
0311 C9 7F              cmp     #$7f            ;if not DEL
0313 D0 F5              bne     5.b             ;or return, go wait for these
0315 20 3BC0            jsr     prtext          ;if DEL
0318 08200800           fcc     $08,$20,$08,0   ;remove character
031C 4C F102            jmp     2.b             ;and go get new command
031F 8A         6       txa                     ;get index in A
0320 0A                 asla                    ;multiply by two
0321 AA                 tax                     ;get new index in X
0322 A9 02              lda     #(nextcmd-1)>>8 ;get high byte of return
0324 48                 pha                     ;address
0325 A9 E4              lda     #(nextcmd-1)&255 ;get lo too
0327 48                 pha                     ;and complete command return address
0328 BD 8F0E            lda     cmdadr+1,x      ;get high byte of command address
032B 48                 pha                     ;on stack
032C BD 8E0E            lda     cmdadr,x        ;get lo too
032F 48                 pha
0330 4C 2FC0            jmp     crlf            ;start on next line and execute command
                ;
                ;*** Command ?: show help text
                ;
0333            qstcmd                          ;print help text and exit
                ;
                ;*** Print helptext
                ;
0333 20 3BC0    qtextpr jsr     prtext          ;print help text
0336 0D54686520         fcc     '\rThe following commands are available:'
033B 666F6C6C6F
0340 77696E6720
0345 636F6D6D61
034A 6E64732061
034F 7265206176
0354 61696C6162
0359 6C653A
035C 0D                 fcc     '\r'
035D 0D42202D20         fcc     '\rB - Set RAM & EPROM address boundaries.'
0362 5365742052
0367 414D202620
036C 4550524F4D
0371 2061646472
0376 6573732062
037B 6F756E6461
0380 726965732E
0385 0943202D20         fcc     $09,'C - Calculate RAM checksum.'
038A 43616C6375
038F 6C61746520
0394 52414D2063
0399 6865636B73
039E 756D2E
03A1 0D44202D20         fcc     '\rD - Set EPROM type.'
03A6 5365742045
03AB 50524F4D20
03B0 747970652E
03B5 0909094520         fcc     $09,$09,$09,'E - Check if EPROM is empty.'
03BA 2D20436865
03BF 636B206966
03C4 204550524F
03C9 4D20697320
03CE 656D707479
03D3 2E
03D4 0D49202D20         fcc     '\rI - Set address increment.'
03D9 5365742061
03DE 6464726573
03E3 7320696E63
03E8 72656D656E
03ED 742E
03EF 09094D202D         fcc     $09,$09,'M - Call MON65 monitor.'
03F4 2043616C6C
03F9 204D4F4E36
03FE 3520D6F6E
0403 69746F722E
0408 0D50202D20         fcc     '\rP - T + program EPROM + V.'
040D 54202B2070
0412 726F677261
0417 6D20455052
041C 4F4D202B20
0421 562E
0423 090951202D         fcc     $09,$09,'Q - Quit program.'
0428 2051756974
042D 2070726F67
0432 72616D2E
0436 0D52202D20         fcc     '\rR - Read EPROM contents into RAM.'
043B 5265616420
0440 4550524F4D
0445 20636F6E74
044A 656E747320
044F 696E746F20
0454 52414D2E
0458 0953202D20         fcc     $09,'S - Show EPROM contents directly.'
045D 53686F7720
0462 4550524F4D
0467 20636F6E74
046C 656E747320
0471 6469726563
0476 746C792E
047A 0D54202D20         fcc     '\rT - Test if EPROM is programmable.'
```

```
047F 5465737420
0484 6966204550
0489 524F4D2069
048E 732070726F
0493 6772616D6D
0498 61626C652E
049D 0956202D20          fcc     $09,'V - Verify EPROM with RAM.'
04A2 5665726966
04A7 7920455052
04AC 4F4D207769
04B1 7468205241
04B6 4D2E
04B8 0D3F202D20          fcc     '\r? - Show this list.'
04BD 53686F7720
04C2 7468697320
04C7 6C6973742E
04CC 0D00                fcc     '\r',0
04CE 60                  rts                     ;and exit
                         ;
                         ;*** Command M: call MON65 monitor
                         ;
04CF 4C 0090     mcmd    jmp     mon             ;call monitor
                         ;
                         ;*** Command Q: exit to DOS
                         ;
04D2 68          qcmd    pla                     ;remove
04D3 68                  pla                     ;return address
04D4 A9 0C               lda     #$0c            ;clear screen
04D6 4C 23C0             jmp     output          ;and exit to DOS
                         ;
                         ;*** Command D: select EPROM type
                         ;
04D9 20 3BC0     dcmd    jsr     prtext          ;print message
04DC 0D2B203D20          fcc     '\r+ = next type, - = previous type, CR = select.\r',0
04E1 6E65787420
04E6 7479706520
04EB 202D203D20
04F0 7072657669
04F5 6F75732074
04FA 7970652C20
04FF 4352203D20
0504 73656C6563
0509 742E0D00
050D AE 2C0F     1       ldx     tabind          ;start with current type
0510 8E 2C0F     2       stx     tabind          ;save X
0513 BD A80E             lda     typtab,x        ;get flags
0516 8D 2B0F             sta     eptyp           ;set type
0519 A2 37               ldx     #55             ;move cursor
051B A0 19               ldy     #25             ;to column 55
051D 20 24F0             jsr     crsadr          ;on statusline
0520 20 32C0             jsr     spa             ;print a space
0523 20 F40C             jsr     prtyp           ;print type
0526 20 27F0             jsr     crsback         ;move cursor to previous position
0529 20 A00C             jsr     settyp          ;reflect type in VIA
052C 20 20C0     3       jsr     input           ;get input
052F AE 2C0F             ldx     tabind          ;get X back
0532 C9 2B               cmp     #'+'            ;if not plus sign
0534 D0 09               bne     4.f             ;test for minus
0536 E8                  inx                     ;else point to next type
0537 E0 14               cpx     #ntyp           ;if not past maximum
0539 D0 D5               bne     2.b             ;go set type
053B A2 00               ldx     #0              ;else point to first type
053D F0 D1               beq     2.b             ;and set it
053F C9 2D       4       cmp     #'-'            ;if not minus sign
0541 D0 07               bne     5.f             ;go test for return
0543 CA                  dex                     ;else point to previous type
0544 10 CA               bpl     2.b             ;if gone through zero
0546 A2 13               ldx     #ntyp-1         ;point to last type
0548 D0 C6               bne     2.b             ;and show that
054A C9 0D       5       cmp     #$0d            ;if not return
054C D0 DE               bne     3.b             ;ignore character
054E 20 21F0             jsr     staton          ;set normal status line
0551 4C 100C             jmp     showbnd         ;exit and show boundaries
                         ;
                         ;*** Command B: show and alter boundaries
                         ;
0554 20 100C     bcmd    jsr     showbnd         ;show boundaries
0557 20 2FC0             jsr     crlf            ;skip a line
055A 20 3BC0     1       jsr     prtext          ;print text
055D 4E65772045          fcc     'New EPROM start address: ',0
0562 50524F4D20
0567 7374617274
056C 2061646472
0571 6573733A20
0576 00
0577 20 580D             jsr     hexin           ;read address
057A 90 06               bcc     2.f             ;if error
057C 20 B60D             jsr     illpr           ;print message
057F 4C 5A05             jmp     1.b             ;and ask for new address
0582 C9 1B       2       cmp     #$1b            ;if ESCape
0584 F0 3F               beq     bex2            ;use defaults
0586 C0 00               cpy     #0              ;if return only
0588 F0 11               beq     pendin          ;leave current start, ask for EPROM end
058A AD 2D0F             lda     temp            ;else get lo
058D AC 2E0F             ldy     temp+1          ;and hi
0590 8D 220F             sta     pstart          ;set start
0593 8C 230F             sty     pstart+1        ;address
0596 20 120A             jsr     epsztst         ;test if within EPROM size
0599 90 41               bcc     epaderr         ;if not, report error
059B 20 3BC0     pendin  jsr     prtext          ;else print text
059E 4E65772045          fcc     'New EPROM  end  address: ',0
05A3 50524F4D20
05A8 20656E6420
05AD 2061646472
05B2 6573733A20
05B7 00
05B8 20 580D             jsr     hexin           ;read address
05BB 90 06               bcc     3.f             ;if error
05BD 20 B60D             jsr     illpr           ;print message
05C0 4C 9B05             jmp     pendin          ;and ask for new address
05C3 C9 1B       3       cmp     #$1b            ;if ESCape
05C5 F0 63       bex2    beq     bex3            ;use defaults
05C7 C0 00               cpy     #0              ;if CR only
05C9 F0 35               beq     rsrtin          ;ask for RAM start
05CB AD 2D0F             lda     temp            ;else get lo
05CE AC 2E0F             ldy     temp+1          ;and hi
05D1 8D 260F             sta     pend            ;set end
05D4 8C 270F             sty     pend+1          ;address
05D7 20 120A             jsr     epsztst         ;test if within EPROM size
05DA B0 24               bcs     rsrtin          ;if so, get RAM start
```

```
05DC 20 3BC0    epaderr jsr     prtext          ;else print text
05DF 0D1B694550         fcc     '\r',$1b,'iEPROM',0
05E4 524F4D00
05E8 20 3BC0    raderr  jsr     prtext
05EB 2073697A65         fcc     ' size exceded.',$1b,'n\r',0
05F0 2065786365
05F5 6465642E1B
05FA 6E0D00
05FD 4C 5A05            jmp     1.b             ;and ask for new EPROM addresses
0600 20 3BC0    rsrtin  jsr     prtext          ;else print text
0603 4E65772020         fcc     'New  RAM  start address: ',0
0608 52414D2020
060D 7374617274
0612 2061646472
0617 6573733A20
061C 00
061D 20 580D            jsr     hexin           ;read address
0620 90 06              bcc     4.f             ;if error
0622 20 B60D            jsr     illpr           ;print message
0625 4C 0006            jmp     rsrtin          ;and ask for new address
0628 C9 1B      4       cmp     #$1b            ;if ESCape,
062A F0 10      bex3    beq     rendshw         ;use defaults
062C C0 00              cpy     #0              ;if CR only
062E F0 0C              beq     rendshw         ;show RAM end address
0630 AD 2D0F            lda     temp            ;else get lo
0633 AC 2E0F            ldy     temp+1          ;and hi
0636 8D 1E0F            sta     rstart          ;set start
0639 8C 1F0F            sty     rstart+1        ;address
063C 20 230A    rendshw jsr     rendset         ;calculate RAM end address
063F B0 9B              bcs     epaderr         ;if negative EPROM size, report error
0641 C9 90              cmp     #ramsize>>8     ;if RAM exceeded
0643 90 0D              bcc     bex             ;
0645 20 3BC0            jsr     prtext          ;print message
0648 0D1B695241         fcc     '\r',$1b,'iRAM',0
064D 4D00
064F 4C E805            jmp     raderr          ;and ask for new addresses
0652 4C 100C    bex     jmp     showbnd         ;else exit showing new boundaries
                ;
                ;*** Command I: enter new address increment
                ;
0655 20 3BC0    icmd    jsr     prtext          ;print prompt
0658 0D4E657720         fcc     '\rNew address increment: ',0
065D 6164647265
0662 737320696E
0667 6372656D65
066C 6E743A2000
0671 20 20C0    1       jsr     input           ;get key
0674 C9 0D              cmp     #$0d            ;if return
0676 F0 1D              beq     3.f             ;leave old increment
0678 C9 31              cmp     #'1'            ;if one
067A F0 08              beq     incv            ;or
067C C9 32              cmp     #'2'            ;two
067E F0 04              beq     incv            ;or
0680 C9 34              cmp     #'4'            ;four, go set increment
0682 D0 ED              bne     1.b             ;if not one of these, get new input
0684 20 23C0    incv    jsr     output          ;echo input
0687 29 07              and     #%00000111      ;remove ASCII
0689 48                 pha                     ;save increment on stack
068A 20 20C0    2       jsr     input           ;get input
068D C9 0D              cmp     #$0d            ;if not return
068F D0 0A              bne     4.f             ;go test for DEL
0691 68                 pla                     ;else get increment back
0692 8D 280F            sta     pinc            ;and store new increment
0695 20 2FC0    3       jsr     crlf            ;start on new line
0698 4C 3C06            jmp     rendshw         ;then go show new addresses
069B C9 7F      4       cmp     #$7f            ;if not DEL
069D D0 EB              bne     2.b             ;go get new input
069F 68                 pla                     ;else restore stack
06A0 20 3BC0            jsr     prtext          ;erase input
06A3 08200800           fcc     $08,$20,$08,0
06A7 4C 7106            jmp     1.b             ;and get new increment
                ;
                ;*** Command C: calculate RAM checksum
                ;
06AA 20 BE0B    ccmd    jsr     clcks           ;clear checksum
06AD 20 C70B            jsr     setrst          ;set RAM address
06B0 A0 00      1       ldy     #0              ;get offset
06B2 B1 80              lda     [rcurr],y       ;get RAM data
06B4 20 DD0A            jsr     addchks         ;add to checksum
06B7 20 D20B            jsr     chkend          ;if at end
06BA F0 06              beq     2.f             ;exit
06BC 20 E20B            jsr     radrinc         ;else get next RAM address
06BF 4C B006            jmp     1.b             ;and loop
06C2 4C B708    2       jmp     prchkr          ;else print checksum and exit
                ;
                ;*** Command P: T + program EPROM + V
                ;
06C5 20 3BC0    pcmd    jsr     prtext          ;print message
06C8 0D42697420         fcc     '\rBit Check..... ',0
06CD 4368656636B
06D2 2E2E2E2E2E
06D7 2000
06D9 A9 C0              lda     #$c0            ;flag
06DB 8D 2F0F            sta     rdver           ;programmability check
06DE 20 8D0A            jsr     rdep            ;go test EPROM
06E1 90 0F              bcc     pok1            ;if no error, go program
06E3 20 3BC0            jsr     prtext          ;else print message
06E6 1B69466169         fcc     $1b,'iFail.',$1b,'n\r',0
06EB 6C2E1B6E0D
06F0 00
06F1 60                 rts                     ;and abort command
06F2 20 DF0D    pok1    jsr     vccon           ;switch Vcc back on
06F5 20 3BC0            jsr     prtext          ;start programming, print message
06F8 4F4B0D5072         fcc     'OK\rProgramming... $00',$08,$08,0
06FD 6F67726156
0702 6D696E672E
0707 2E2E202430
070C 3008080
0710 20 E20D            jsr     vppon           ;switch Vpp on
0713 AD B3E7            lda     brkvector       ;get user
0716 8D 300F            sta     brksav          ;break vector
0719 AD B4E7            lda     brkvector+1     ;in save
071C 8D 310F            sta     brksav+1        ;area
071F 78                 sei                     ;no interrupts
0720 A9 08              lda     #ownbrk>>8      ;point user
0722 8D B4E7            sta     brkvector+1     ;break
0725 A9 7B              lda     #ownbrk&255     ;to
0727 8D B3E7            sta     brkvector       ;own routine
072A 58                 cli                     ;allow interrupts
072B BA                 tsx                     ;get stack pointer
```

```
072C 8E 320F            stx     savstk          ;save it
072F AD 2B0F            lda     eptyp           ;get EPROM type
0732 0A                 asla                    ;move algorithm bits
0733 0A                 asla                    ;to C and N flags
0734 90 0C              bcc     pint            ;if inteligent or 50ms, go do that
0736 30 2E              bmi     pee             ;if EEPROM, go do that
0738 20 DC0D            jsr     vcc6            ;else raise Vcc to 6 volts
073B 20 630B            jsr     prgqck          ;go program quick pulse
073E 90 2A              bcc     pv              ;if OK, go verify
0740 B0 0A              bcs     perror          ;else print message and exit
0742 10 23      pint    bpl     p50             ;if 50ms, go do that
0744 20 DC0D            jsr     vcc6            ;else raise Vcc to 6 Volts
0747 20 0A0B            jsr     prgint          ;and go program inteligent
074A 90 1E              bcc     pv              ;if no error, go verify
074C 20 EE0D    perror  jsr     zerotxt         ;if error, zero Volt on all pins
074F 20 3BC0            jsr     prtext          ;print message
0752 0D1B695072         fcc     '\r',$1b,'iProgram Fail.',$1b,'n\r',0
0757 6F67726165
075C 2046616963
0761 2E1B6E0D00
0766 60         pee     rts                     ;and do next command
0767 20 EA0A    p50     jsr     prg50           ;go program 50ms
076A 20 0E0E    pv      jsr     vppoff          ;switch programming voltage off
076D 20 A00B            jsr     oldbrk          ;restore user break vector
0770 20 3BC0            jsr     prtext          ;print message
0773 0D56657269         fcc     '\rVerifying..... ',0
0778 6679696E67
077D 2E2E2E2E2E
0782 2000
0784 A9 80              lda     #$80            ;flag
0786 8D 2F0F            sta     rdver           ;verify
0789 20 8D0A            jsr     rdep            ;verify EPROM
078C B0 2E              bcs     verror          ;if error go display
078E 20 3BC0            jsr     prtext          ;else print message
0791 4F4B0D0700         fcc     'OK\r',$07,0
0796 4C B708            jmp     prchkr          ;print checksum and get next command
                ;
                ;*** Command V: verify EPROM with RAM
                ;(Entered at label vcmd)
0799 20 3BC0    vok     jsr     prtext          ;print message
079C 4550524F4D         fcc     'EPROM equals RAM.\r',0
07A1 2065717561
07A6 6C73205241
07AB 4D2E0D00
07AF 4C B708            jmp     prchkr          ;print checksum and get next command
07B2 A9 80      vcmd    lda     #$80            ;flag
07B4 8D 2F0F            sta     rdver           ;verify
07B7 20 8D0A            jsr     rdep            ;go verify EPROM
07BA 90 DD              bcc     vok             ;if no error, exit
07BC 20 3BC0    verror  jsr     prtext          ;else print message
07BF 1B69566572         fcc     $1b,'iVerify fail.',$1b,'n',$09
07C4 6966792066
07C9 61696C2E1B
07CE 6E09
07D0 2853706163         fcc     '(Space = next failure, CR = abort command)\r\r',0
07D5 65203D206E
07DA 6578742066
07DF 61696C7572
07E4 652C204352
07E9 203D206162
07EE 6F72742063
07F3 6F6D6D616E
07F8 64290D0D00
07FD 20 3BC0    verloop jsr     prtext
0800 4550524F4D         fcc     'EPROM address: $',0
0805 2061646472
080A 6573733A20
080F 2400
0811 AD 240F            lda     pcurr           ;get lo
0814 AC 250F            ldy     pcurr+1         ;and hi
0817 20 780E            jsr     yahex           ;print EPROM address
081A 20 3BC0            jsr     prtext          ;print message
081D 094550524F         fcc     $09,'EPROM data: $',0
0822 4D20646174
0827 613A202400
082C AD 2D0F            lda     temp            ;get EPROM data
082F 20 3BC0            jsr     prbyte          ;print in hex
0832 20 3BC0            jsr     prtext          ;print message
0835 0920205241         fcc     $09,'  RAM data: $',0
083A 4D20646174
083F 613A202400
0844 AD 2E0F            lda     temp+1          ;get RAM data
0847 20 3BC0            jsr     prbyte          ;print too
084A 20 2FC0            jsr     crlf            ;start on next line
084D 20 20C0    1       jsr     input           ;get a key
0850 C9 0D              cmp     #$0d            ;if carriage return
0852 F0 26              beq     2.f             ;abort command
0854 C9 20              cmp     #$20            ;if not space
0856 D0 F5              bne     1.b             ;ignore and get new key
0858 20 620A            jsr     rdep1           ;else continue verify
085B B0 A0              bcs     verloop         ;if failure, print data and addresses
085D 20 3BC0            jsr     prtext          ;else report
0860 0D4E6F206D         fcc     '\rNo more differences.\r',0
0865 6F72652064
086A 6966666572
086F 656E636573
0874 2E0D00
0877 20 EE0D            jsr     zerotxt         ;switch all supplies off
087A 60         2       rts                     ;and do next command
                ;
                ;*** User break routine
                ;
087B 20 0E0E    ownbrk  jsr     vppoff          ;switch Vpp off
087E 20 EE0D            jsr     zerotxt         ;and set all pins at zero volt
0881 20 A00B            jsr     oldbrk          ;restore break vector
0884 20 3BC0            jsr     prtext          ;print
0887 0D0D071B69         fcc     '\r\r',7,$1b,'iProgramming aborted by ^C.',$1b,'n\r',0
088C 5072GF6772
0891 616D6D696E
0896 6720616F6F
089B 7274656420
08A0 6279205E43
08A5 2E1B6E0D00
08AA AE 320F            ldx     savstk          ;get original stack pointer
08AD 9A                 txs                     ;as current value
08AE 60                 rts                     ;and get new command
                ;
                ;*** Command R: read EPROM into RAM
                ;
08AF A9 00      rcmd    lda     #0              ;flag
```

```
08B1 8D 2F0F            sta     rdver           ;read
08B4 20 8D0A            jsr     rdep            ;go read EPROM
08B7 20 F80B    prchkr  jsr     prchks          ;print checksum
08BA 4C 2FC0            jmp     crlf            ;start on new line and exit
                ;
                ;*** Command E: check if EPROM is empty
                ;
08BD 20 7B0A    ecmd    jsr     etcmd           ;test if EPROM empty
08C0 20 030A            jsr     cnot            ;print NOT if error
08C3 20 3BC0    eok     jsr     prtext          ;print message
08C6 656D707479         fcc     'empty.\r',0
08CB 2E0D00
08CE 60                 rts                     ;then get next command
                ;
                ;*** Command T: check if EPROM is programmable
                ;
08CF 20 7B0A    tcmd    jsr     etcmd           ;test if EPROM empty
08D2 90 EF              bcc     eok             ;if no error, report empty
08D4 A9 C0              lda     #$c0            ;else
08D6 8D 2F0F            sta     rdver           ;flag programmability check
08D9 20 620A            jsr     rdep1           ;proceed with check
08DC 20 030A            jsr     cnot            ;print NOT if error
08DF 20 3BC0            jsr     prtext          ;print message
08E2 70726F6772         fcc     'programmable.\r',0
08E7 616D6D6162
08EC 6C652E0D00
08F1 60                 rts                     ;then get next command
                ;
                ;*** Command S: show EPROM contents without loading RAM
                ;
08F2 AD 220F    scmd    lda     pstart          ;get EPROM start
08F5 AC 230F            ldy     pstart+1        ;address in YA
08F8 8D 240F            sta     pcurr           ;and move it
08FB 8C 250F            sty     pcurr+1         ;to current address
08FE 20 7E09    1       jsr     sdump           ;dump 16 lines
0901 20 3BC0            jsr     prtext          ;print banner
0904 0D2B203D20         fcc     '\r+ = next 16 lines, - = previous 16 lines, '
0909 6E65787420
090E 3136206C69
0913 6E65732C20
0918 2D203D2070
091D 726576696F
0922 7573203136
0927 206C696E65
092C 732C20
092F 4352203D20         fcc     'CR = quit command\r',0
0934 7175697420
0939 636F6D6D61
093E 6E640D00
0942 20 20C0    2       jsr     input           ;get input
0945 C9 2B              cmp     #'+'            ;if next 16 lines
0947 D0 12              bne     3.f
0949 20 EA0C            jsr     getsize         ;get EPROM size
094C CD 250F            cmp     pcurr+1         ;if EPROM address hi lower
094F B0 AD              bcs     1.b             ;go dump
0951 A9 00              lda     #0              ;else set
0953 8D 240F            sta     pcurr           ;EPROM address
0956 8D 250F            sta     pcurr+1         ;to zero
0959 F0 A3              beq     1.b             ;and go dump
095B C9 2D      3       cmp     #'-'            ;if not previous 16 lines
095D D0 1A              bne     5.f             ;go check for return
095F 38                 sec                     ;else prepare for subtract
0960 AD 250F            lda     pcurr+1         ;get EPROM address hi
0963 E9 02              sbc     #$02            ;go 2 pages back
0965 8D 250F            sta     pcurr+1         ;set new address
0968 B0 94              bcs     1.b             ;if not through zero, go dump
096A 20 EA0C            jsr     getsize         ;else get EPROM size in A
096D 30 03              bmi     4.f             ;if 27512, set zero lo
096F 8D 250F            sta     pcurr+1         ;else set new high address
0972 A9 00      4       lda     #0              ;and zero
0974 8D 240F            sta     pcurr           ;low EPROM address
0977 F0 B5              beq     1.b             ;and go dump
0979 C9 0D      5       cmp     #$0d            ;if not return
097B D0 C5              bne     2.b             ;get new input
097D 60                 rts                     ;else exit
                ;
                ;*** Subroutine,sdump: hexdump 16 lines of EPROM data
                ;
097E 20 DF0D    sdump   jsr     vccon           ;switch Vcc on
0981 20 1F0E            jsr     dinoe           ;databus to inputs and apply OE
0984 A9 10              lda     #16             ;16 lines
0986 8D 2E0F            sta     temp+1          ;to do
0989 AD 240F            lda     pcurr           ;get current
098C AC 250F            ldy     pcurr+1         ;address
098F 20 B80B            jsr     movcurr         ;move it to EPROM
0992 20 2FC0            jsr     crlf            ;start on new line
0995 AD 90E1    1       lda     para            ;get EPROM address lo
0998 48                 pha                     ;on stack
0999 AD 92E1            lda     parb            ;get hi
099C 48                 pha                     ;on stack too
099D A9 10              lda     #16             ;16 bytes per line
099F 8D 2D0F            sta     temp            ;
09A2 20 720E            jsr     prepadr         ;print current EPROM address
09A5 20 32C0            jsr     spa             ;print
09A8 20 32C0            jsr     spa             ;two spaces
09AB AD 8FE1    2       lda     varanh          ;get EPROM data
09AE 20 38C0            jsr     prbyte          ;print in hex
09B1 20 32C0            jsr     spa             ;print a space
09B4 AD 2D0F            lda     temp            ;if
09B7 C9 09              cmp     #9              ;at byte
09B9 D0 03              bne     3.f             ;number 8
09BB 20 32C0            jsr     spa             ;print another space
09BE 20 EF0B    3       jsr     epadinc         ;adapt EPROM address
09C1 CE 2D0F            dec     temp            ;adapt counter
09C4 D0 E5              bne     2.b             ;if not 16 bytes done, loop
09C6 68                 pla                     ;else put EPROM address hi back
09C7 8D 92E1            sta     parb            ;on EPROM
09CA 68                 pla                     ;get lo too
09CB 8D 90E1            sta     para            ;and complete address
09CE 20 32C0            jsr     spa             ;print another space
09D1 A9 10              lda     #16             ;get
09D3 8D 2D0F            sta     temp            ;counter
09D6 AD 8FE1    4       lda     varanh          ;get EPROM data
09D9 C9 20              cmp     #$20            ;if not a control character
09DB B0 0A              bcs     5.f             ;print byte directly
09DD 48                 pha                     ;else save the byte
09DE 20 3BC0            jsr     prtext          ;enable
09E1 1B4600             fcc     $1b,'F',0       ;graphics display
09E4 68                 pla                     ;get byte back
09E5 09 60              ora     #$60            ;make it printable
```

```
09E7 20 23C0    5       jsr     output          ;print it
09EA 20 3BC0            jsr     prtext          ;disable graphics display
09ED 1B4700             fcc     $1b,'G',0       ;
09F0 20 EF0B            jsr     epadinc         ;adapt EPROM address
09F3 CE 2D0F            dec     temp            ;adapt counter
09F6 D0 DE              bne     4.b             ;if not all done, loop
09F8 20 2FC0            jsr     crlf            ;else start on new line
09FB CE 2E0F            dec     temp+1          ;adapt line counter
09FE D0 95              bne     1.b             ;if 16 lines done, exit
0A00 4C EE0D            jmp     zerotxt         ;else exit with Vcc off
                ;
                ;*** Subroutine cnot: print NOT if carry set
                ;
0A03 90 0C      cnot    bcc     nonot           ;if no error, exit
0A05 20 3BC0            jsr     prtext          ;else print message
0A08 1B694E4F54         fcc     $1b,'iNOT',$1b,'n ',0
0A0D 1B6E2000
0A11 60         nonot   rts                     ;and exit
                ;
                ;*** Subroutine epsztst: test if address in YA is within EPROM size
                ;Carry clear means error
0A12 8C 2E0F    epsztst sty     temp+1          ;save hi byte
0A15 AD 2B0F            lda     eptyp           ;get current EPROM type
0A18 29 07              and     #%00000111      ;get index
0A1A AA                 tax                     ;in X
0A1B BD 080F            lda     epsize,x        ;get size hi
0A1E 38                 sec                     ;prepare for subtract
0A1F ED 2E0F            sbc     temp+1          ;subtract address hi
0A22 60                 rts                     ;set carry if address valid
                ;
                ;*** Subroutine rendset: calculate and set RAM end address
                ;
0A23 38         rendset sec                     ;prepare for subtract
0A24 AD 260F            lda     pend            ;get EPROM end lo
0A27 ED 220F            sbc     pstart          ;and get EPROM size
0A2A 8D 2D0F            sta     temp            ;in temp
0A2D AD 270F            lda     pend+1          ;get EPROM end, hi
0A30 ED 230F            sbc     pstart+1        ;calculate size hi
0A33 8D 2E0F            sta     temp+1          ;store it
0A36 90 28              bcc     epader          ;if size negative, exit with error
0A38 30 26              bmi     epader          ;if over $8000, exit with error
0A3A AD 280F            lda     pinc            ;get address increment
0A3D 4A                 lsra                    ;get number of shifts
0A3E AA                 tax                     ;in X
0A3F F0 0B              beq     setrend         ;if no shift, set RAM end
0A41 0E 2D0F    1       asl     temp            ;else shift
0A44 2E 2E0F            rol     temp+1          ;size
0A47 B0 17              bcs     epader          ;if through $FFFF, flag error
0A49 CA                 dex                     ;else count shifts
0A4A D0 F5              bne     1.b             ;and loop if not completed
0A4C 18         setrend clc                     ;prepare for add
0A4D AD 2D0F            lda     temp            ;get new size lo
0A50 6D 1E0F            adc     rstart          ;add RAM start lo
0A53 8D 200F            sta     rend            ;and set end lo
0A56 AD 2E0F            lda     temp+1          ;get size hi
0A59 6D 1F0F            adc     rstart+1        ;add start hi
0A5C 8D 210F            sta     rend+1          ;and complete end address
0A5F 24                 fcc     $24             ;skip next instruction
0A60 38         epader  sec                     ;flag error
0A61 60                 rts                     ;then exit
                ;
                ;*** Subroutine rdep1: read/verify/check EPROM with current address
                ;
0A62 20 DF0D    rdep1   jsr     vccon           ;switch Vcc on
0A65 AC 250F            ldy     pcurr+1         ;get current address hi
0A68 AD 240F            lda     pcurr           ;and lo
0A6B 20 B80B            jsr     movcurr         ;set current EPROM address
0A6E 20 D20B            jsr     chkend          ;if at end address
0A71 D0 02              bne     1.f
0A73 18                 clc                     ;flag no error
0A74 60                 rts                     ;and exit
0A75 20 DF0B    1       jsr     adrinc          ;else point to next address
0A78 4C 930A            jmp     r1              ;and proceed
                ;
                ;*** Subroutine etcmd: test if EPROM empty
                ;
0A7B A9 40      etcmd   lda     #$40            ;flag
0A7D 8D 2F0F            sta     rdver           ;empty test
0A80 20 3BC0            jsr     prtext          ;print message
0A83 4550524F4D         fcc     'EPROM is ',0   ;then:
0A88 2069732000
                ;
                ;*** Subroutine rdep: read/check/verify EPROM into/with RAM
                ;
0A8D 20 DF0D    rdep    jsr     vccon           ;switch Vcc on
0A90 20 AF0B            jsr     movstrt         ;set begin address on EPROM
0A93 20 1F0E    r1      jsr     dinoe           ;set databus to inputs and apply OE
0A96 20 650E            jsr     wait01          ;wait 100 us
0A99 AD 8FE1    1       lda     voranh          ;get EPROM data in A
0A9C A0 00              ldy     #0              ;get offset zero
0A9E 2C 2F0F            bit     rdver           ;get read/verify flag
0AA1 30 04              bmi     2.f             ;if verify, skip storage
0AA3 70 0A              bvs     3.f             ;if empty test, compare with $FF
0AA5 91 80              sta     [rcurr],y       ;else move data to RAM
0AA7 70 0C      2       bvs     5.f             ;if programmability check, do that
0AA9 D1 80      4       cmp     [rcurr],y       ;if verify fail
0AAB D0 23              bne     9.f             ;exit
0AAD F0 0E              beq     6.f             ;else calculate checksum
0AAF C9 FF      3       cmp     #$ff            ;if not empty
0AB1 D0 1D              bne     9.f             ;exit
0AB3 F0 0B              beq     7.f             ;else do not calculate checksum
0AB5 B1 80      5       lda     [rcurr],y       ;programmability check, get RAM data
0AB7 2D 8FE1            and     voranh          ;and with EPROM data, if bit(s) go zero
0ABA 4C A90A            jmp     4.b             ;fail verify and exit
0ABD 20 DD0A    6       jsr     addchks         ;add to checksum
0AC0 20 D20B    7       jsr     chkend          ;test if at end address
0AC3 F0 06              beq     8.f             ;if so, stop reading
0AC5 20 DF0B            jsr     adrinc          ;else adapt addresses
0AC8 4C 990A            jmp     1.b             ;and loop
0ACB 20 EE0D    8       jsr     zerotxt         ;zero volt on textool socket
0ACE 18                 clc                     ;flag no error
0ACF 60                 rts                     ;and exit
0AD0 8D 2D0F    9       sta     temp            ;store EPROM data
0AD3 B1 80              lda     [rcurr],y       ;get RAM data
0AD5 8D 2E0F            sta     temp+1          ;save too
0AD8 20 EE0D            jsr     zerotxt         ;zero volt on textool socket
0ADB 38                 sec                     ;flag error
0ADC 60                 rts                     ;and exit
                ;
                ;*** Subroutine addchks: add A to checksum
```

```
                        ;
0ADD 18         addchks clc                     ;else prepare for add
0ADE 6D 290F            adc     chksum          ;add data to checksum
0AE1 8D 290F            sta     chksum          ;store new checksum lo
0AE4 90 03              bcc     1.f             ;if page crossed
0AE6 EE 2A0F            inc     chksum+1        ;adapt hi byte
0AE9 60         1       rts                     ;and exit
                        ;
                        ;*** Program EPROM, 50 ms
                        ;
0AEA 20 AF0B    prg50   jsr     movstrt         ;set start address on EPROM
0AED 20 D00D            jsr     noedato         ;switch databus to outputs
0AF0 A0 00      1       ldy     #0              ;get offset
0AF2 B1 80              lda     [rcurr],y       ;get RAM data
0AF4 C9 FF              cmp     #$ff            ;if empty state
0AF6 F0 06              beq     2.f             ;skip programming
0AF8 8D 81E1            sta     vora            ;else set data onto EPROM
0AFB 20 240E            jsr     pulse50         ;apply program pulse and print address
0AFE 20 D20B    2       jsr     chkend          ;if at end address,
0B01 F0 06              beq     3.f             ;exit
0B03 20 DF0B            jsr     adrinc          ;else get next address
0B06 4C F00A            jmp     1.b             ;and loop
0B09 60         3       rts                     ;else exit
                        ;
                        ;*** Program EPROM, Inteligent mode
                        ;
0B0A 20 AF0B    prgint  jsr     movstrt         ;set start address on EPROM
0B0D A0 00      1       ldy     #0              ;clear
0B0F 8C 1D0F            sty     pcnt            ;pulse counter
0B12 B1 80              lda     [rcurr],y       ;get RAM data
0B14 C9 FF              cmp     #$ff            ;if empty state
0B16 F0 3B              beq     6.f             ;skip programming
0B18 8D 8FE1            sta     voranh          ;else get data onto EPROM
0B1B 48         2       pha                     ;and on stack
0B1C 20 D00D            jsr     noedato         ;set databus to outputs
0B1F 20 3D0E            jsr     pulse1          ;apply a 1ms pulse and print address hi
0B22 EE 1D0F            inc     pcnt            ;increment pulse counter
0B25 AD 0302            lda     avar            ;get maximum count
0B28 CD 1D0F            cmp     pcnt            ;if maximum exceeded
0B2B 90 33              bcc     perr            ;exit with error
0B2D 20 1F0E            jsr     dinoe           ;else set databus to inputs and apply OE
0B30 68                 pla                     ;get RAM data back
0B31 EA                 nop                     ;allow EPROM to
0B32 EA                 nop                     ;switch to read mode
0B33 CD 8FE1            cmp     voranh          ;if data not the same
0B36 D0 E3              bne     2.b             ;apply another pulse
0B38 20 D00D            jsr     noedato         ;set databus to outputs
0B3B AD 1D0F            lda     pcnt            ;get number of pulses so far
0B3E 18                 clc                     ;prepare for add
0B3F AE 0402            ldx     bvar            ;get overprogram factor
0B42 F0 0F              beq     6.f             ;if zero, check for end address
0B44 CA         3       dex                     ;else adapt count
0B45 F0 05              beq     4.f             ;if count calculated, proceed
0B47 6D 1D0F            adc     pcnt            ;add number of pulses
0B4A D0 F8              bne     3.b             ;and loop (branch always)
0B4C AA         4       tax                     ;get over program pulse length in X
0B4D 20 3D0E    5       jsr     pulse1          ;apply program pulse
0B50 CA                 dex                     ;adapt count
0B51 D0 FA              bne     5.b             ;if not all done, loop
0B53 20 D20B    6       jsr     chkend          ;if at end address,
0B56 F0 06              beq     7.f             ;exit
0B58 20 DF0B            jsr     adrinc          ;else get next address
0B5B 4C 0D0B            jmp     1.b             ;and loop
0B5E 18         7       clc                     ;flag no error
0B5F 60                 rts                     ;and exit
0B60 68         perr    pla                     ;correct stack
0B61 38                 sec                     ;flag error
0B62 60                 rts                     ;and exit
                        ;
                        ;*** Program EPROM, Quick Pulse
                        ;
0B63 20 AF0B    prgqck  jsr     movstrt         ;get start address on EPROM
0B66 A0 00      1       ldy     #0              ;clear
0B68 8C 1D0F            sty     pcnt            ;pulse counter
0B6B B1 80              lda     [rcurr],y       ;get RAM data
0B6D C9 FF              cmp     #$ff            ;if empty state
0B6F F0 22              beq     3.f             ;skip programming
0B71 8D 8FE1            sta     voranh          ;onto EPROM
0B74 48         2       pha                     ;and on stack
0B75 20 D00D            jsr     noedato         ;set databus to outputs
0B78 20 5E0E            jsr     pulse01         ;apply a 0.1 ms pulse and print address
0B7B EE 1D0F            inc     pcnt            ;increment pulse counter
0B7E A9 19              lda     #25             ;get maximum count
0B80 CD 1D0F            cmp     pcnt            ;if maximum exceeded
0B83 90 0B              bcc     perr            ;exit with error
0B85 20 1F0E            jsr     dinoe           ;else set databus to inputs and apply OE
0B88 68                 pla                     ;get RAM data back
0B89 EA                 nop                     ;allow EPROM to
0B8A EA                 nop                     ;switch to read mode
0B8B CD 8FE1            cmp     voranh          ;if data not the same
0B8E D0 E4              bne     2.b             ;apply another pulse
0B90 20 D00D            jsr     noedato         ;else set databus to outputs
0B93 20 D20B    3       jsr     chkend          ;if at end address,
0B96 F0 06              beq     4.f             ;exit
0B98 20 DF0B            jsr     adrinc          ;else get next address
0B9B 4C 660B            jmp     1.b             ;and loop
0B9E 18         4       clc                     ;flag no error
0B9F 60                 rts                     ;and exit
                        ;
                        ;*** Subroutine oldbrk: restore original user break vector
                        ;
0BA0 78         oldbrk  sei                     ;no interrupts
0BA1 AD 300F            lda     brksav          ;restore
0BA4 8D B3E7            sta     brkvector       ;old
0BA7 AD 310F            lda     brksav+1        ;break
0BAA 8D B4E7            sta     brkvector+1     ;vector
0BAD 58                 cli                     ;allow interrupts
0BAE 60                 rts                     ;and exit
                        ;
                        ;*** Subroutine movstrt: set first EPROM address on EPROM
                        ;
0BAF 20 C70B    movstrt jsr     setrst          ;set RAM start address
0BB2 AD 220F            lda     pstart          ;get EPROM start lo
0BB5 AC 230F            ldy     pstart+1        ;get hi too
0BB8 8D 90E1    movcurr sta     pora            ;set address
0BBB 8C 92E1            sty     porb            ;on EPROM
0BBE A9 00      clcks   lda     #0              ;clear
0BC0 8D 290F            sta     chksum          ;checksum
0BC3 8D 2A0F            sta     chksum+1        ;and
0BC6 60                 rts                     ;exit
```

```
                        ;
                        ;*** Subroutine setrst: move RAM start to current RAM address
                        ;
0BC7 AD 1E0F    setrst  lda     rstart          ;get RAM start lo
0BCA AC 1F0F            ldy     rstart+1        ;get hi too
0BCD 85 80              sta     rcurr           ;in current address
0BCF 84 81              sty     rcurr+1         ;and complete RAM address
0BD1 60                 rts
                        ;
                        ;*** Subroutine chkend: test if at end address
                        ;
0BD2 A5 81      chkend  lda     rcurr+1         ;get RAM address hi
0BD4 CD 210F            cmp     rend+1          ;if not equal to end hi
0BD7 D0 05              bne     1.f             ;exit with Z=0
0BD9 A5 80              lda     rcurr           ;else get lo
0BDB CD 200F            cmp     rend            ;set Z if address is equal
0BDE 60         1       rts                     ;and exit
                        ;
                        ;*** Subroutine adrinc: set EPROM and RAM address to next value
                        ;
0BDF 20 EF0B    adrinc  jsr     epadinc         ;adapt EPROM address
0BE2 A5 80      radrinc lda     rcurr           ;get current RAM address lo
0BE4 18                 clc                     ;prepare for add
0BE5 6D 280F            adc     pinc            ;add address increment
0BE8 85 80              sta     rcurr           ;if page crossed
0BEA 90 02              bcc     2.f             ;adapt
0BEC E6 81              inc     rcurr+1         ;hi as well
0BEE 60         2       rts                     ;then exit
                        ;
                        ;*** Subroutine epadinc: increment EPROM address
                        ;
0BEF EE 90E1    epadinc inc     porа            ;adapt EPROM address lo
0BF2 D0 03              bne     1.f             ;if page crossed
0BF4 EE 92E1            inc     porb            ;adapt hi
0BF7 60         1       rts                     ;and exit
                        ;
                        ;*** Subroutine prchks: print checksum in hex
                        ;
0BF8 20 3BC0    prchks  jsr     prtext          ;print text
0BFB 4368656368         fcc     'Checksum= $',0
0C00 73756D3D20
0C05 2400
0C07 AC 2A0F            ldy     chksum+1        ;get checksum hi
0C0A AD 290F            lda     chksum          ;and lo
0C0D 4C 780E            jmp     yahex           ;print checksum in hex
                        ;
                        ;*** Subroutine showbnd: show boundaries
                        ;
0C10 20 3BC0    showbnd jsr     prtext          ;then print text
0C13 0D41646472         fcc     '\rAddress increment = ',0
0C18 6573732069
0C1D 6E6372656D
0C22 656E74203D
0C27 2000
0C29 AD 280F            lda     pinc            ;get address increment
0C2C 20 35C0            jsr     hnout           ;print address increment
0C2F 20 3BC0            jsr     prtext          ;print string
0C32 0D4550524F         fcc     '\rEPROM start: $',0
0C37 4D20737461
0C3C 72743A2024
0C41 00
0C42 AC 230F            ldy     pstart+1        ;get EPROM start address hi
0C45 AD 220F            lda     pstart          ;get lo as well
0C48 20 780E            jsr     yahex           ;print address
0C4B 20 3BC0            jsr     prtext          ;print string
0C4E 0909524144         fcc     $09,$09,'RAM start: $',0
0C53 2073746172
0C58 743A202400
0C5D AC 1F0F            ldy     rstart+1        ;get RAM start address hi
0C60 AD 1E0F            lda     rstart          ;get lo as well
0C63 20 780E            jsr     yahex           ;print address
0C66 20 3BC0            jsr     prtext          ;print string
0C69 0D4550524F         fcc     '\rEPROM  end : $',0
0C6E 4D20206E6E
0C73 64203A2024
0C78 00
0C79 AC 270F            ldy     pend+1          ;get RAM start address hi
0C7C AD 260F            lda     pend            ;get lo as well
0C7F 20 780E            jsr     yahex           ;print address
0C82 20 3BC0            jsr     prtext          ;print string
0C85 0909524144         fcc     $09,$09,'RAM  end : $',0
0C8A 2020656E64
0C8F 203A202400
0C94 AC 210F            ldy     rend+1          ;get RAM start address hi
0C97 AD 200F            lda     rend            ;get lo as well
0C9A 20 780E            jsr     yahex           ;print address
0C9D 4C 2FC0            jmp     crlf            ;then print CRLF and exit
                        ;
                        ;*** Subroutine settyp: set EPROM type and default boundaries
                        ;
0CA0 20 EA0C    settyp  jsr     getsize         ;get size and index
0CA3 48                 pha                     ;save size
0CA4 AD 80E1            lda     vorb            ;get control lines
0CA7 29 F8              and     #%11111000      ;clear type
0CA9 8D 80E1            sta     vorb
0CAC 8A                 txa                     ;get type back
0CAD 0D 80E1            ora     vorb            ;add to control lines
0CB0 8D 80E1            sta     vorb            ;and set new type
0CB3 68                 pla                     ;get size, hi
0CB4 29 7F              and     #$7f            ;remove MSbit ($8000 is max. size)
0CB6 8D 270F            sta     pend+1          ;store it
0CB9 A0 FF              ldy     #$ff            ;get lo
0CBB 8C 260F            sty     pend            ;and complete size
0CBE C8                 iny                     ;Y=0
0CBF 8C 220F            sty     pstart          ;set $0000
0CC2 8C 230F            sty     pstart+1        ;as EPROM start
0CC5 8C 1E0F            sty     rstart          ;and $1000
0CC8 A0 10              ldy     #$10            ;as
0CCA 8C 1F0F            sty     rstart+1        ;RAM start
0CCD A9 01              lda     #1              ;set default address step
0CCF 8D 280F            sta     pinc            ;to 1
0CD2 20 230A            jsr     rendset         ;set RAM end address
0CD5 2C 2B0F            bit     eptyp           ;get Vpp flag in N
0CD8 10 07              bpl     vppa            ;and:
                        ;
                        ;*** Set Vpp B
                        ;
0CDA AD 8CE1    vppb    lda     vpcr            ;get CB2 control
0CDD 09 20              ora     #%00100000      ;set CB2
0CDF D0 05              bne     1.f             ;to one and exit
```

```
                        ;
                        ;*** Set Vpp A
                        ;
0CE1 AD 8CE1    vppa    lda     vpcr            ;get CB2 control
0CE4 29 DF              and     #%11011111      ;reset CB2
0CE6 8D 8CE1    1       sta     vpcr            ;to zero
0CE9 60                 rts                     ;and exit
                        ;
                        ;*** Subroutine getsize: get EPROM size hi in A
                        ;
0CEA AD 2B0F    getsize lda     eptyp           ;get current type
0CED 29 07              and     #%00000111      ;convert to index
0CEF AA                 tax                     ;in X
0CF0 BD 080F            lda     epsize,x        ;get size
0CF3 60                 rts                     ;and exit
                        ;
                        ;*** Subroutine prtyp: print current EPROM type and algorithm
                        ;
0CF4 AD 2B0F    prtyp   lda     eptyp           ;get EPROM type in use
0CF7 29 07              and     #%00000111      ;get type bits only
0CF9 48                 pha                     ;save number
0CFA 0A                 asla                    ;multiply by two
0CFB 0A                 asla                    ;by four
0CFC 8D 2D0F            sta     temp            ;save A
0CFF 68                 pla                     ;get number back
0D00 6D 2D0F            adc     temp            ;calculate index
0D03 AA                 tax                     ;move index to X
0D04 A0 05              ldy     #5              ;get character counter
0D06 BD BC0E    1       lda     eptext,x        ;get character of name
0D09 20 23C0            jsr     output          ;print on screen
0D0C E8                 inx                     ;adapt index
0D0D 88                 dey                     ;and character counter
0D0E D0 F6              bne     1.b             ;if all characters printed, loop
0D10 A9 20              lda     #' '            ;get a space
0D12 2C 2B0F            bit     eptyp           ;if VppA
0D15 10 02              bpl     2.f             ;print a space
0D17 A9 41              lda     #'A'            ;else get an A
0D19 20 23C0    2       jsr     output          ;print it
0D1C 20 32C0            jsr     spa             ;print
0D1F AD 2B0F            lda     eptyp           ;get EPROM type
0D22 29 18              and     #%00011000      ;clear unnecessary bits
0D24 4A                 lsra                    ;convert to index
0D25 AA                 tax                     ;move index to X
0D26 A0 04              ldy     #4              ;get character counter
0D28 BD E40E    4       lda     vpptext,x       ;get text character
0D2B 20 23C0            jsr     output          ;print it
0D2E E8                 inx                     ;adapt index
0D2F 88                 dey                     ;if not all printed,
0D30 D0 F6              bne     4.b             ;loop
0D32 20 3BC0            jsr     prtext          ;print string
0D35 562000             fcc     'V ',0
0D38 AD 2B0F            lda     eptyp           ;get type again
0D3B 29 60              and     #%01100000      ;get algorithm bits
0D3D 4A                 lsra                    ;get bits
0D3E 4A                 lsra                    ;into
0D3F 4A                 lsra                    ;position
0D40 8D 2D0F            sta     temp            ;save fourfold
0D43 4A                 lsra                    ;convert
0D44 4A                 lsra                    ;to text number
0D45 6D 2D0F            adc     temp            ;add fourfold
0D48 AA                 tax                     ;get index to text
0D49 A0 05              ldy     #5              ;get character counter
0D4B BD F40E    5       lda     algtext,x       ;get character
0D4E 20 23C0            jsr     output          ;print it
0D51 E8                 inx                     ;adapt index
0D52 88                 dey                     ;if not all characters printed
0D53 D0 F6              bne     5.b             ;loop
0D55 4C 32C0            jmp     spa             ;print a space and exit
                        ;
                        ;*** Subroutine hexin: read hexadecimal input into temp
                        ;CR only --> C=0, Y=0
                        ;ESC, CR --> C=0, Y=1
                        ;Illegal hex digit --> C=1
0D58 20 29C0    hexin   jsr     bufin           ;get input until CR
0D5B 84 80              sty     rcurr           ;save length
0D5D C0 00              cpy     #0              ;if return only
0D5F F0 09              beq     escfnd          ;exit with C=0, Y=0
0D61 C9 1B              cmp     #$1b            ;if ESC
0D63 F0 05              beq     escfnd          ;flag that
0D65 20 6C0D            jsr     hexnum          ;else get hexadecimal input
0D68 B0 01              bcs     errout          ;if error, print that
0D6A 18         escfnd  clc                     ;flag no error
0D6B 60         errout  rts                     ;
                        ;
                        ;*** Subroutine hexnum: read hexadecimal input into temp
                        ;(C=1 on exit illegal hex digit found)
0D6C A0 00      hexnum  ldy     #0              ;clear
0D6E 8C 2D0F            sty     temp            ;work
0D71 8C 2E0F            sty     temp+1          ;area
0D74 84 81              sty     rcurr+1         ;store index
0D76 A4 81      1       ldy     rcurr+1         ;get buffer index
0D78 C4 80              cpy     rcurr           ;if at end
0D7A F0 21              beq     crout           ;exit
0D7C 20 2CC0            jsr     getbuf          ;else get input character
0D7F E6 81              inc     rcurr+1         ;point to next
0D81 20 41C0            jsr     loupch          ;convert to upper case
0D84 20 9F0D            jsr     aschex          ;else convert input character to nibble
0D87 B0 15              bcs     illhex          ;if error, exit
0D89 A2 03              ldx     #3              ;get bit counter
0D8B 0E 2D0F    2       asl     temp            ;move
0D8E 2E 2E0F            rol     temp+1          ;nibble to next nibble
0D91 CA                 dex                     ;if not all bits done,
0D92 10 F7              bpl     2.b             ;loop
0D94 0D 2D0F            ora     temp            ;add nibble in A to inl
0D97 8D 2D0F            sta     temp            ;and store new value
0D9A 4C 760D            jmp     1.b             ;and get next digit
0D9D 18         crout   clc                     ;C=0 (=OK)
0D9E 60         illhex  rts                     ;and exit
                        ;
                        ;*** Subroutine aschex: convert ASCII hex digit to nibble
                        ;
0D9F C9 30      aschex  cmp     #'0'            ;if below zero
0DA1 90 11              bcc     nvalid          ;check for control characters
0DA3 C9 3A              cmp     #'9'+1          ;if digit,
0DA5 90 0A              bcc     valid           ;go add to result
0DA7 C9 41              cmp     #'A'            ;if not a valid
0DA9 90 09              bcc     nvalid          ;letter, exit
0DAB C9 47              cmp     #('F'+1)        ;if beyond F
0DAD B0 05              bcs     nvalid          ;exit with C=1
0DAF 69 09              adc     #9              ;else add nine if a letter
```

```
0DB1 29 0F      valid   and     #%00001111      ;remove ASCII bits
0DB3 24                 fcb     $24             ;skip next instruction
0DB4 38         nvalid  sec                     ;set carry if error
0DB5 60                 rts                     ;and exit
                ;
                ;*** subroutine illpr: print illegal hex digit
                ;
0DB6 20 3BC0    illpr   jsr     prtext          ;print message
0DB9 070D496C6C         fcc     $07,'\rIllegal hex digit\r\r',0
0DBE 6567616C20
0DC3 6865782064
0DC8 696769740D
0DCD 0D00
0DCF 60                 rts
                ;
                ;*** Remove OE and switch databus to outputs
                ;
0DD0 20 E50D    noedato jsr     nooe            ;set EPROM in tri-state, and:
                ;
                ;*** Set databus to outputs
                ;
0DD3 A9 FF      dataout lda     #$ff            ;get value
0DD5 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Set databus to inputs
                ;
0DD6 A9 00      datain  lda     #0              ;get value
0DD8 8D 83E1            sta     vddra           ;set port
0DDB 60                 rts                     ;and exit
                ;
                ;*** Set Vcc to 6 Volts
                ;
0DDC A9 08      vcc6    lda     #%00001000      ;get mask
0DDE 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Switch Vcc on
                ;
0DDF A9 10      vccon   lda     #%00010000      ;get mask
0DE1 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Switch Vpp on
                ;
0DE2 A9 20      vppon   lda     #%00100000      ;get mask
0DE4 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Set OE high
                ;
0DE5 A9 40      nooe    lda     #%01000000      ;get mask
0DE7 0D 80E1            ora     vorb            ;set bit
0DEA 8D 80E1            sta     vorb            ;in VIA
0DED 60                 rts                     ;then exit
                ;
                ;*** Zero volt on all textool pins
                ;
0DEE AD 90E1    zerotxt lda     pora            ;get address lo
0DF1 8D 240F            sta     pcurr           ;save it
0DF4 AD 92E1            lda     porb            ;and hi
0DF7 8D 250F            sta     pcurr+1         ;too
0DFA A0 00              ldy     #0              ;set all
0DFC 8C 90E1            sty     pora            ;addresslines
0DFF 8C 92E1            sty     porb            ;to zero
0E02 8C 8FE1            sty     voranh          ;zero databus on switch to outputs
0E05 20 0E0E            jsr     vppoff          ;switch Vpp off
0E08 20 D00D            jsr     noedato         ;set databus to outputs, no OE, then:
                ;
                ;*** Switch Vcc off
                ;
0E0B A9 EF      vccoff  lda     #%11101111      ;get mask
0E0D 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Switch Vpp off
                ;
0E0E A9 DF      vppoff  lda     #%11011111      ;get mask
0E10 20 180E            jsr     rsbit           ;switch Vpp off, then:
                ;
                ;*** Set Vcc to 5 Volts
                ;
0E13 A9 F7      vcc5    lda     #%11110111      ;get mask
0E15 2C                 fcc     $2c             ;skip next instruction
                ;
                ;*** Set OE low
                ;
0E16 A9 BF      oeon    lda     #%10111111      ;get mask
0E18 2D 80E1    rsbit   and     vorb            ;reset bit
0E1B 8D 80E1            sta     vorb            ;in VIA
0E1E 60                 rts                     ;then exit
                ;
                ;*** Set databus to inputs and apply OE
                ;
0E1F 20 D60D    dinoe   jsr     datain          ;set databus to inputs
0E22 F0 F2              beq     oeon            ;and apply OE
                ;
                ;*** Apply a 50 ms program pulse
                ;
                pulse50 if      2==3            ;if speed is 3 MHz

0E24 20 270E            jsr     pulse5a         ;apply two pulses of 50/2 ms

0E27 A9 50      pulse5a lda     #50000&255      ;get lo
0E29 A0 C3              ldy     #50000>>8       ;and hi
0E2B 8D 86E1    pulsex  sta     vt1ll           ;write value lo
0E2E 8C 85E1            sty     vt1ch           ;write hi as well and start pulse
0E31 20 720E            jsr     prepadr         ;print EPROM address
0E34 20 3BC0            jsr     prtext          ;then print two backspaces
0E37 080800             fcc     $08,$08,0
0E3A 4C 520E            jmp     pulsbs          ;and wait until time-out
                ;
                ;*** Apply a 1 ms program pulse
                ;
0E3D A9 D0      pulse1  lda     #(1000*2)&255   ;get lo
0E3F A0 07              ldy     #(1000*2)>>8    ;and hi
0E41 8D 86E1    pulsx   sta     vt1ll           ;write value lo
0E44 8C 85E1            sty     vt1ch           ;write hi as well and start pulse
0E47 AD 90E1            lda     pora            ;get EPROM address lo
0E4A D0 0C              bne     pulsew          ;if not on page start, go wait for time-out
0E4C AD 92E1            lda     porb            ;get EPROM address hi
0E4F 20 38C0            jsr     prbyte          ;print it in hex
0E52 20 3BC0    pulsbs  jsr     prtext          ;then print two backspaces
0E55 080800             fcc     $08,$08,0
0E58 2C 8DE1    pulsew  bit     vifr            ;get flag register bit 6 in V
```

```
0E5B 50 F8              bvc     pulsew          ;and wait for time-out
0E5D 60                 rts                     ;then exit
                        ;
                        ;*** Apply a 0.1 ms program pulse
                        ;
0E5E A9 C8      pulse01 lda     #(100*2)&255    ;get lo
0E60 A0 00              ldy     #(100*2)>>8     ;and hi
0E62 4C 410E            jmp     pulsx           ;and apply pulse
                        ;
                        ;*** Wait 100 us
                        ;
0E65 A9 C8      wait01  lda     #(100*2)&255    ;get lo
0E67 A0 00              ldy     #(100*2)>>8     ;and hi
0E69 8D 86E1            sta     vt1ll           ;write value lo
0E6C 8C 85E1            sty     vt1ch           ;write hi as well and start pulse
0E6F 4C 580E            jmp     pulsew          ;and wait for time-out
                        ;
                        ;*** Subroutine prepadr: print current EPROM address in hex
                        ;
0E72 AC 92E1    prepadr ldy     porb            ;get hi byte of EPROM address
0E75 AD 90E1            lda     pora            ;get lo too and:
                        ;
                        ;*** Subroutine yahex: print YA in hex
                        ;
0E78 48         yahex   pha                     ;save A
0E79 98                 tya                     ;get hi in A
0E7A 20 38C0            jsr     prbyte          ;print in hex
0E7D 68                 pla                     ;get lo back
0E7E 4C 38C0            jmp     prbyte          ;print too, then exit
                        ;
                        ;*** Table of available commands
                        ;
0E81 42         cmds    fcc     'B'             ;set boundaries
0E82 43                 fcc     'C'             ;calculate RAM checksum
0E83 44                 fcc     'D'             ;select EPROM type
0E84 45                 fcc     'E'             ;test if EPROM empty
0E85 49                 fcc     'I'             ;set address increment
0E86 4D                 fcc     'M'             ;call monitor
0E87 50                 fcc     'P'             ;program EPROM
0E88 51                 fcc     'Q'             ;quit, exit to DOS
0E89 52                 fcc     'R'             ;read EPROM into RAM
0E8A 53                 fcc     'S'             ;show EPROM contents on screen
0E8B 54                 fcc     'T'             ;test if EPROM programmable
0E8C 56                 fcc     'V'             ;verify EPROM with RAM
0E8D 3F                 fcc     '?'             ;show help file
          000D  ncmds   equ     $-cmds          ;number of commands
                        ;
                        ;*** Table of addresses of commands
                        ;
0E8E 5305       cmdadr  fdb     bcmd-1
0E90 A906               fdb     ccmd-1
0E92 D804               fdb     dcmd-1
0E94 BC08               fdb     ecmd-1
0E96 5406               fdb     icmd-1
0E98 CE04               fdb     mcmd-1
0E9A C406               fdb     pcmd-1
0E9C D104               fdb     qcmd-1
0E9E AE08               fdb     rcmd-1
0EA0 F108               fdb     scmd-1
0EA2 CE08               fdb     tcmd-1
0EA4 B107               fdb     vcmd-1
0EA6 3203               fdb     qstcmd-1
                        ;
                        ;*** Table of available types
                        ;
                        ;bits 0-2: type
                        ;bits 3-4: 0=25 V, 1=21 V, 2=12.5 V, 3 unused
                        ;bits 5-6: 0=50 ms algorithm
                        ;          1=inteligent
                        ;          2=quick pulse
                        ;          3=EEPROM
                        ;bit  7   : 0=VppA, 1=VppB
0EA8 00         typtab  fcb     %00000000       ; 2716, 25V, 50 ms
                ;       fcb     %10001000       ; 2516A, 21V, 50 ms
0EA9 01                 fcb     %00000001       ; 2732, 25V, 50 ms
0EAA 89                 fcb     %10001001       ; 2732A, 21V, 50 ms
0EAB 0A                 fcb     %00001010       ; 2764, 21V, 50 ms
0EAC 2A         epdflt  fcb     %00101010       ; 2764, 21V, inteligent
0EAD B2                 fcb     %10110010       ; 2764A, 12.5V, inteligent
0EAE D2                 fcb     %11010010       ; 2764A, 12.5V, quick pulse
0EAF 0B                 fcb     %00001011       ;27128, 21V, 50 ms
0EB0 2B                 fcb     %00101011       ;27128, 21V, inteligent
0EB1 B3                 fcb     %10110011       ;27128A, 12.5V, inteligent
0EB2 D3                 fcb     %11010011       ;27128A, 12.5V, quick pulse
0EB3 2C                 fcb     %00101100       ;27256, 21V, inteligent
0EB4 B4                 fcb     %10110100       ;27256A, 12.5V, inteligent
0EB5 D4                 fcb     %11010100       ;27256A, 12.5V, quick pulse
0EB6 35                 fcb     %00110101       ;27512, 12.5V, inteligent
0EB7 55                 fcb     %01010101       ;27512, 12.5V, quick pulse
0EB8 06                 fcb     %00000110       ; 2532, 25V, 50 ms
0EB9 8E                 fcb     %10001110       ; 2532A, 21V, 50 ms
0EBA 07                 fcb     %00000111       ; 2564, 25V, 50 ms
0EBB 8F                 fcb     %10001111       ; 2564A, 21V, 50 ms
          0014  ntyp    equ     $-typtab
          0004  dfltyp  equ     epdflt-typtab
```

```
                        ;
                        ;*** Table of EPROM type texts
                        ;
0EBC 2032373136 eptext  fcc     ' 2716'
0EC1 2032373332         fcc     ' 2732'
0EC6 2032373634         fcc     ' 2764'
0ECB 3237313238         fcc     '27128'
0ED0 3237323536         fcc     '27256'
0ED5 3237353132         fcc     '27512'
0EDA 2032353332         fcc     ' 2532'
0EDF 2032353634         fcc     ' 2564'
                        ;
                        ;*** Table of Vpp texts
                        ;
0EE4 20203235   vpptext fcc     '  25'
0EE8 20203231           fcc     '  21'
0EEC 31322E35           fcc     '12.5'
0EF0 2020203F           fcc     '   ?'
                        ;
                        ;*** Table of algorithm texts
                        ;
0EF4 2035306D73 algtext fcc     ' 50ms'
0EF9 496E74656C         fcc     'Intel'
0EFE 5175636368         fcc     'Quick'
0F03 2045452020         fcc     ' EE  '
                        ;
                        ;*** Size table, hi bytes only
                        ;
0F08 07         epsize  fcb     $07             ; 2716
0F09 0F                 fcb     $0F             ; 2732
0F0A 1F                 fcb     $1F             ; 2764
0F0B 3F                 fcb     $3F             ;27128
0F0C 7F                 fcb     $7F             ;27256
0F0D FF                 fcb     $FF             ;27512
0F0E 0F                 fcb     $0F             ; 2532
0F0F 1F                 fcb     $1F             ; 2564
                        ;
                        ;*** DOS command: load MON65
                        ;
0F10 4C4F20533A monstr  fcc     'LO S:MONITOR',0
0F15 4D4F4E4954
0F1A 4F5200
                        ;
                        ;*** Various variables
                        ;
0F1D            pcnt    res     1               ;number of pulses done
0F1E            rstart  res     2               ;start address in RAM
0F20            rend    res     2               ;end address in RAM
0F22            pstart  res     2               ;start address of EPROM
0F24            pcurr   res     2               ;current EPROM address
0F26            pend    res     2               ;maximum address in EPROM
0F28            pinc    res     1               ;address increment
0F29            chksum  res     2               ;checksum over EPROM data
0F2B            eptyp   res     1               ;current EPROM type
0F2C            tabind  res     1               ;index in type table
0F2D            temp    res     2               ;temporary work area
0F2F            rdver   res     1               ;read/verify flag
0F30            brksav  res     2               ;room for user break vector save
0F32            savstk  res     1               ;save area for stack pointer



          0200          end     ep
```

global labels

addchks	0ADD	adrinc	0BDF	algtext	0EF4	aschex	0D9F	avar	0203
bcmd	0554	bex	0652	bex2	05C5	bex3	062A	brksav	0F30
brkvector	E7B3	bufin	C029	bvar	0204	ccmd	06AA	chkend	0BD2
chksum	0F29	clcks	0BBE	cmdadr	0E8E	cmds	0E81	cnot	0A03
crlf	C02F	crout	0D9D	crsadr	F024	crsback	F027	datain	0DD6
dataout	0DD3	dcmd	04D9	dfltyp	0004	dinoe	0E1F	doscom	C006
ecmd	08BD	eok	08C3	ep	0200	ep1	0272	epader	0A60
epaderr	05DC	epadinc	0BEF	epbase	E180	epdflt	0EAC	epsize	0F08
epsztst	0A12	eptext	0EBC	eptyp	0F2B	errout	0D6B	escfnd	0D6A
etcmd	0A7B	getbuf	C02C	getsize	0CEA	hexin	0D58	hexnum	0D6C
hnout	C035	icmd	0655	illhex	0D9E	illpr	0DB6	incv	0684
input	C020	loupch	C041	mcmd	04CF	mon	9000	monstr	0F10
movcurr	0BB8	movstrt	0BAF	ncmds	000D	nextcmd	02E5	noedato	0DD0
nonot	0A11	nooe	0DE5	ntyp	0014	nvalid	0DB4	oeon	0E16
oldbrk	0BA0	output	C023	ownbrk	087B	p50	0767	pcmd	06C5
pcnt	0F1D	pcra	E191	pcrb	E193	pcurr	0F24	pee	0766
pend	0F26	pendin	059B	perr	0B60	perror	074C	pia	E190
pinc	0F28	pint	0742	pok1	06F2	pora	E190	porb	E192
prbyte	C038	prchkr	0BB7	prchks	0BF8	prepadr	0E72	prg50	0AEA
prgint	0B0A	prgqck	0B63	prtext	C03B	prtyp	0CF4	pstart	0F22
pulsbs	0E52	pulse01	0E5E	pulse1	0E3D	pulse50	0E24	pulse5a	0E27
pulsew	0E58	pulsex	0E2B	pulsx	0E41	pv	076A	qcmd	04D2
qstcmd	0333	qtextpr	0333	r1	0A93	raderr	05E8	radrinc	0BE2
ramsize	9000	rcmd	0BAF	rcurr	0080	rdep	0A8D	rdep1	0A62
rdver	0F2F	release	3030	rend	0F20	rendset	0A23	rendshw	063C
rev	3156	rsbit	0E18	rsrtin	0600	rstart	0F1E	savstk	0F32
scmd	08F2	sdump	097E	setrend	0A4C	setrst	0BC7	settyp	0CA0
showbnd	0C10	spa	C032	statoff	F01B	staton	F021	stattog	E701
tabind	0F2C	tcmd	08CF	temp	0F2D	typtab	0EA8	vacr	E18B
valid	0DB1	vcc5	0E13	vcc6	0DDC	vccoff	0E0B	vccon	0DDF
vcmd	07B2	vddra	E183	vddrb	E182	verloop	07FD	verror	07BC
via	E180	vier	E18E	vifr	E18D	vok	0799	vora	E181
voranh	E18F	vorb	E180	vper	E18C	vppa	0CE1	vppb	0CDA
vppoff	0E0E	vppon	0DE2	vpptext	0EE4	vsr	E18A	vt1ch	E185
vt1cl	E184	vt1lh	E187	vt1ll	E186	vt2ch	E189	vt2cl	E188
wait01	0E65	yahex	0E78	zerotxt	0DEE				

Errors detected: 0