```
; ELBUG monitor
; Elektor SC/MP issue January 1978 (German)
;

; memory layout
; 0000-05f0 Elbug ROM (3*MM5204)
; 0600-06ff unused, may be 256 bytes RAM
; 0700-0707 hex display
; 0708-070f keyboard
; 07xx      mirrors of display and keyboard
; 0800-0fff up to 2k RAM
; 0f00-0fff must be present, monitor RAM and stack area

; I/O ports
hexio:      .= x'700          ; right most 7 seg digit
hexio1:     .= x'701          ; typical pos of P2 pointing to display
hexio6:     .= x'706
hexio7:     .= x'707          ; left most 7 seg digit
hexin:      .= x'708          ; keyboard input

; keyboard input word is
; 7  6  5  4  3  2  1  0
; |  |  |  |  +--+--+--+------ hex keycode (0-f)
; |  +--+--+------------------ command key (0-7)
; +--------------------------- is key pressed (0,1)

;           command keys
            ky_run = x'f0
            ky_mod = x'e0
            ky_sub = x'd0
            ky_cas = x'c0
            ky_blk = x'b0
            ky_cpu = x'a0
            ky_dn  = x'90
            ky_up  = x'80
            ky_a   = x'fa
            ky_e   = x'fe
            ky_s   = x'f5
            ky_1   = x'f1
            ky_2   = x'f2

stktop:     .= x'fe0
            dta = 0           ; data field for display
            addrl = 1         ; temporary address
            addrh = 2

            digit4 = 2        ; save for address digits for display
            digit3 = 3
            digit2 = 4
            digit1 = 5
            digit0 = 6
            digitm = 7

            blkcnt = 5        ; block count in CAS read and write
            cksum = 6         ; checksum buffer in CAS read and write
            bytesv = 7        ; byte save for CAS write
            bitcnt = 8        ; bit count for CAS read and write
            bitdly = 9        ; bit delay for CAS read write
            bitspd = x'a      ; delay for bit for CAS read and write
            casl = x'b        ; CAS start address
            cash = x'c
            adflg = x'10      ; flag: use read address from tape
```

```
                    spdsav = x'14     ; save for CAS read speed
                    speed = x'15      ; speed for CAS data transfer

                    key7sg = 7        ; 7 segment value of hex key pressed
                    keycod = 8        ; key code of pressed key
                    keyhex = 9        ; hex key part of pressed key

                    count = x'b       ; digit count in main loop

                    bldifh = x'b      ; block difference in block move
                    bldifl = x'c
                    blendh = x'd      ; block end address in block move
                    blendl = x'e
                    blbegh = x'f      ; block start address in block move
                    blbegl = x'10

                    cpuh = x'd        ; CPU start address
                    cpul = x'e

                    outb = x'f        ; number of bytes to output in display

                    subl = x'13       ; 1st operand for subtraction
                    subh = x'14

                    psaveh = x'16     ; PC save
                    psavel = x'17
                    acsave = x'18     ; AC save

                    spcnt = x'19      ; SP counter for nested subroutines
                    level = x'1a      ; Stack level
                    incall = x'1b     ; in CALL flag
                    callh = x'1c      ; address of caller
                    calll = x'1d
                    sph = x'1e        ; stack pointer
                    spl = x'1f

          top:      .= x'ff0          ; top of RAM

          ; stack frame:
          ;     dw    caller address    ; will be filled with caller by user code
          ;     db    p3h
          ;     db    p3l
          ;     db    p2h
          ;     db    p2l
          ;     db    p1h
          ;     db    p1l
          ;     db    status reg
          ;     db    e reg
          ;     db    accu reg

                    .= x'000
          cold:                       ; cold reset entry
0000 08             nop               ; unusable instr because of PC pre-increment
0001 c415           ldi x'15          ; initialize speed of cassette
0003 c8f1           st stktop+speed
0005 c4e0           ldi l(stktop)     ; init stack pointer
0007 c8f7           st stktop+spl     ; to point to x'0fe0 (stktop)
0009 c40f           ldi h(stacktop)
000b c8f2           st stktop+sph
000d c400           ldi x'00          ; clear number of nested routines
000f c8e9           st stktop+spcnt
0011 c8e9           st stktop+incall  ; clear incall flag
```

```
0013 903d              jmp dowarm       ; skip to warm reset

           ; this is the entry point of the RETURN helper
           ; which pops off the registers from the last stack frame

0015 c0e9  return:     ld stktop+spl    ; load stack ptr
0017 31                xpal p1          ; into P1
0018 c0e5              ld stktop+sp
001a 35                xpah p1

001b c501              ld @1(p1)        ; restore caller
001d c8de              st stktop+callh  ; from stack frame
001f c501              ld @1(p1)
0021 c8db              st stktop+calll

0023 c501              ld @1(p1)        ; restore stack frame p3h
0025 37                xpah p3          ; into p3
0026 c501              ld @1(p1)        ; and also p3l
0028 33                xpal p3          ; into p3

0029 c501              ld @1(p1)        ; restore stack frame p2h
002b 36                xpah p2          ; into p2
002c c501              ld @1(p1)        ; and also p2l
002e 32                xpal p2

002f c501              ld @1(p1)        ; restore stack frame p1h
0031 c8c4              st stktop+psaveh ; into a temporary
0033 c501              ld @1(p1)        ; also p1l
0035 c8c1              st stktop+psavel

0037 c501              ld @1(p1)        ; restore stack frame status
0039 07                cas              ; into status reg
003a c501              ld @1(p1)        ; restore stack frame e
003c 01                xae              ; into e reg
003d c501              ld @1(p1)        ; restore stack frame accu
003f c8b8              st stktop+acsave ; into a temporary

0041 c0b4              ld stktop+psaveh ; get P1 temporary
0043 35                xpah p1          ; xchg with p1h
0044 c8b9              st stktop+sph    ; save new sph
0046 c0b0              ld psavel        ; same for low byte of P1
0048 31                xpal p1          ; xchg with p1l
0049 c8b5              st stktop+spl    ; save new spl

004b b8ad              dld stktop+spcnt ; decrement nesting level
004d c0aa              ld stktop+acsave ; restore accu
004f 3f                xppc p3          ; jump to return address

           ; official entry points
0050 9004  docall:     jmp call         ; skip to call entry
0052 904d  dowarm:     jmp warm         ; skip to warm entry
0054 90bf  doreturn:   jmp return       ; skip to return entry

           ; this is the entry point of the CALL helper
           ; load the address you want to call into
           ; stktop+callh/calll and xppc3 to this call entry
           ; will save the registers to the stack pointed to by stkh/stkl
           ; and prepare return point of P3 to point to doreturn
0056 c8a1  call:       st stktop+acsave ; save accu

0058 c0a6              ld stktop+spl    ; load spl into P3
005a 33                xpal p3          ; xchg with p3l
```

```
005b c89b              st stktop+psavel  ; store p3l into temporary (return to caller)
005d c0a0              ld stktop+sph     ; load sph into P3
005f 37                xpah p3           ; xchg with p3h
0060 c895              st stktop+psaveh  ; store p3h into temporary

0062 c4ff              ldi l(top)        ; xchg top ram address into p1l
0064 31                xpal p1           ;
0065 cffc              st @-4(p3)        ; save p1l into stack frame
0067 c40f              ldi h(top)        ; xchg sp address into p1h
0069 35                xpah p1
006a cfff              st @-1(p3)        ; save p1h into stack frame
006c 01                xae               ; get e reg
006d cb03              st 3(p3)          ; save into stack frame
006f 06                csa               ; get status reg
0070 cb02              st 2(p3)          ; save into stack frame
0072 c1f9              ld -7(p1)         ; get temporary acsave via p1
0074 cb04              st 4(p3)          ; save into stack frame
0076 32                xpal p2           ; get p2l
0077 cfff              st @-1(p3)        ; save into stack frame
0079 36                xpah p2           ; get p2h
007a cfff              st @-1(p3)        ; save into stack frame
007c c1f8              ld -8(p1)         ; get saved p3l via p1
007e cfff              st @-1(p3)        ; save into frame
0080 c1f7              ld -9(p1)         ; get saved p3h via p1
0082 cfff              st @-1(p3)        ; save into frame

0084 c1fe              ld -2(p1)         ; get call addressl via p1
0086 cfff              st @-1(p3)        ; save into stack frame
0088 c1fd              ld -1(p1)         ; get call addressh via p1
008a cfff              st @-1(p3)        ; save into stack frame
008c 37                xpah p3           ; xchg into p3 (sph into accu)
008d c9ff              st -1(p1)         ; store new stack ptr h
008f c1fe              ld -2(p1)         ; get call addressl via p1
0091 33                xpal p3           ; xchg into p3 (spl into accu)
0092 c900              st 0(p1)          ; store new stack ptr l

0094 a9fa              ild -6(p1)        ; increment spcnt via p1
0096 e1fb              xor -5(p1)        ; compare with calllevel
0098 9c04              jnz callit        ; if not zero skip

009a c4ff              ldi x'ff          ; mark: in call, call frame contains data
009c c9fc              st -4(p1)         ; set marker

009e 3f     callit:    xppc p3           ; call address
                                         ; unmodified P3 will return here and does
restore via RETURN
009f 90b3              jmp doreturn      ; restore stackframe

            ; this is the warm start entry point

00a1 c400   warm:      ldi l(hexio)      ; load p1 with IO area
00a3 31                xpal p1
00a4 c407              ldi h(hexio)
00a6 35                xpah p1

00a7 c4e0              ldi l(stktop)     ; load p2 with monitor area
00a9 32                xpal p2
00aa c40f              ldi h(stktop)
00ac 36                xpah p2

00ad c42f              ldi l(elbug)      ; load p3 with 7-segment elbug text
00af 33                xpal p3
```

```
00b0 c401              ldi h(elbug)
00b2 37                xpah p3

00b3 c408              ldi x'08          ; counter for 7 segment displays
00b5 ca0b              st count(p2)      ; into temp counter

           ; this loop will display "..ELBuG "
00b7 c701  elloop:     ld @1(p3)         ; load first 7seg char
00b9 cd01              st @1(p1)         ; put into display (build display "..ELBuG ")
00bb ba0b              dld count(p2)     ; decrement count
00bd 9cf8              jnz elloop        ; loop over 8 chars

00bf c40a              ldi l(waitky)-1   ; preset waitky routine
00c1 ca1d              st calll(p2)      ; in callh/l
00c3 c402              ldi h(waitky)
00c5 ca1c              st callh(p2)

00c7 c40037c455333f    js p3,call        ; run call helper

00ce c480              ldi x'80          ; 7segment period
00d0 cdfd              st @-3(p1)        ; build display "......  "
00d2 cdff              st @-1(p1)
00d4 cdff              st @-1(p1)
00d6 cdff              st @-1(p1)
00d8 c400              ldi x'00          ; empty field (where 'G' was)
00da cdff              st @-1(p1)

00dc c208              ld keycod(p2)     ; get keycode
00de 01                xae               ; save into E
00df 40                lde               ; reload it into accu
00e0 e4e0              xri ky_mod        ; is modify key?
00e2 9853              jz domod          ; yes, skip to modify

00e4 40                lde               ; reload key
00e5 e4f0              xri ky_run        ; is run key?
00e7 9c07              jnz warm1         ; no skip
00e9 c40137c4a0333f    js p3,dorun       ; goto run

00f0 40    warm1:      lde               ; reload key
00f1 e4d0              xri ky_sub        ; is subtract key?
00f3 9c07              jnz warm2         ; no skip
00f5 c40337c4ea333f    js p3,dosub       ; goto subtract

00fc 40    warm2:      lde               ; reload key
00fd e4c0              xri ky_cas        ; is key cassette?
00ff 9c07              jnz warm3
0101 c40237c4f1333f    js p3,docas       ; goto cassette

0108 40    warm3:      lde               ; reload key
0109 e4b0              xri ky_blk        ; is key block transfer?
010b 9c07              jnz warm4
010d c40537c449333f    js p3,doblk       ; goto block transfer

0114 40    warm4:      lde               ; reload key
0115 e4a0              xri ky_cpu        ; is key cpureg?
0117 9c88              jnz warm          ; no loop back
0119 c40437c435333f    js p3,docpu       ; goto cpureg

           ; this is the HEX to 7 segment conversion table
           ; trick: x'0 is x'3f in 7segment, this recycles
           ; the last xppc3 of above js macro, so to load hextable
           ; one must use l(hextbl)-1 actually
```

```
0120 06     hextbl:     .byte x'06        ; '1'
0121 5b                 .byte x'5b        ; '2'
0122 4f                 .byte x'4f        ; '3'
0123 66                 .byte x'66        ; '4'
0124 6d                 .byte x'6d        ; '5'
0125 7d                 .byte x'7d        ; '6'
0126 07                 .byte x'07        ; '7'
0127 7f                 .byte x'7f        ; '8'
0128 6f                 .byte x'6f        ; '9'
0129 77                 .byte x'77        ; 'A'
012a 7c                 .byte x'7c        ; 'b'
012b 58                 .byte x'58        ; 'C'
012c 5e                 .byte x'5e        ; 'd'
012d 79                 .byte x'79        ; 'E'
012e 71                 .byte x'71        ; 'F'

            ; this is the monitor prompt, reverse
012f 00     elbug:      .byte x'00        ; ' '
0130 3d                 .byte x'3d        ; 'G'
0131 1c                 .byte x'1c        ; 'u'
0132 7c                 .byte x'7c        ; 'b'
0133 38                 .byte x'38        ; 'L'
0134 79                 .byte x'79        ; 'E'
0135 80                 .byte x'80        ; '.'
0136 80                 .byte x'80        ; '.'

            ; entry point of MODIFY
0137 c45c   domode:     ldi x'5c          ; p1 points to hexio1
0139 c905               st 5(p1)          ; 'o' into display
013b c454               ldi x'54
013d c906               st 6(p1)          ; 'm' into display

013f c43e               ldi l(getadr)-1   ; load getadr into caller
0141 ca1d               st calll(p2)      ; note: call helper is still in P3,
0143 3f                 xppc p3           ; and callh=02 == h(getadr)

            ; main loop of modify
0144 c201   moloop:     ld addrl(p2)      ; load address into p3
0146 33                 xpal p3
0147 c202               ld addrh(p2)
0149 37                 xpah p3

014a c300               ld 0(p3)          ; read data value from address
014c ca00               st dta(p2)        ; store it into data field
014e c4a0               ldi l(disply)-1   ; display address and data
0150 ca1d               st calll(p2)      ; note; callh=02
0152 c40037c455333f     js p3,call        ; call display address+data

0159 c40a               ldi l(waitky)-1   ; load waitky
015b ca1d               st calll(p2)
015d 3f                 xppc p3           ; call helper

015e c201               ld addrl(p2)      ; get address into p3
0160 33                 xpal p3
0161 c202               ld addrh(p2)
0163 37                 xpah p3

0164 c208               ld kycode(p2)     ; get keycode
0166 e480               xri ky_up         ; is UP key?
0168 980a               jz doup           ; yes, skip
016a e480               xri ky_up         ; undo xor
016c e490               xri ky_dn         ; is DOWN key?
```

```
016e 9c0e               jnz domod3        ; no skip

0170 c7ff               ld @-1(p3)        ; DN: get previous data value (decrement p3)
0172 9002               jmp domod2        ; continue
0174 c701   doup:       ld @1(p3)         ; UP: get next data value (increment p3)

0176 33     domod2:     xpal p3           ; save address
0177 ca01               st addrl(p2)      ; into addrl parameter
0179 37                 xpah p3
017a ca02               st addrh(p2)
017c 90c6               jmp moloop        ; loop modify

017e c207   domod3:     ld key7sg(p2)     ; must be hex key, get 7seg value
0180 c900               st 0(p1)          ; put into display (position 1)
0182 c400               ldi x'00          ; clear other display nibble
0184 c9ff               st -1(p1)
0186 c209               ld keyhex(p2)     ; get hex value
0188 1e                 rr                ; rotate into upper nibble
0189 1e                 rr
018a 1e                 rr
018b 1e                 rr
018c 01                 xae               ; save in E

018d c40037c455333f     js p3,call        ; still waitky preloaded, wait for key
0194 c201               ld addrl(p2)      ; set p3 = address
0196 33                 xpal p3
0197 c202               ld addrh(p2)
0199 37                 xpah p3

019a c209               ld keyhex(p2)     ; get hex value
019c 58                 ore               ; merge with first nibble
019d cb00               st 0(p3)          ; store at memory position
019f 90a3               jmp moloop        ; loop to modify
                        ; this routine is left by RESET

            ; entry point of RUN
01a1 c450   dorun:      ldi x'50          ; p1 points to hexio1
01a3 c906               st 6(p1)          ; 'r' in display
01a5 c41c               ldi x'1c          ; 'u' in display
01a7 c905               st 5(p1)

01a9 c43e               ldi l(getadr)-1   ; call get address
01ab ca1d               st calll(p2)      ; note callh=02
01ad c40037c455333f     js p3,call        ; call helper

01b4 c40a               ld l(waitky)-1    ; call waitkey
01b6 ca1d               st calll(p2)
01b8 3f                 xppc p3           ; call helper

01b9 c201               ld addrl(p2)      ; set p3 = address
01bb 33                 xpal p3
01bc c202               ld addrh(p2)
01be 37                 xpah p3

01bf c7ff               ld @-1(p3)        ; adjust address (SC/MP is preincrement PC)

01c1 c450               ldi x'50          ; 'r' in display
01c3 c900               st 0(p1)
01c5 c41c               ldi x'1c          ; 'u' in display
01c7 c9ff               st -1(p1)

01c9 3f                 xppc p3           ; execute routine
```

```
01ca c40f37c4ff333f     js p3,cold        ; goto cold start

            ; cassette helper routine to read a byte
01d1 c215   rbyte:      ld speed(p2)      ; get speed constant
01d3 1c                 sr                ; divide by 2
01d4 ca14               st spdsav(p2)     ; store it

01d6 c4ff   rbyte1:     ldi x'ff          ; preset all ones
01d8 01                 xae               ; into E
01d9 19                 sio               ; shift bit in
01da 40                 lde               ; get current value
01db 9402               jp rbyte2         ; start bit seen?
01dd 90f7               jmp rbyte1        ; no wait

01df c4ff   rbyte2:     ldi x'ff          ; preset all ones again
01e1 01                 xae               ; into E
01e2 c214               ld spdsav(p2)     ; get speed count
01e4 ca0a               st spdcnt(p2)     ; store in temporary

01e6 ba0a   rbyte3:     dld spdcnt(p2)    ; decrement speed delay
01e8 9cfc               jnz rbyte3        ; wait

01ea c408               ldi x'08          ; number of bits
01ec ca08               st bitcnt(p2)     ; save into bit counter

rbyte4:
01ee c215   rbyte4:     ld speed(p2)      ; get bit delay
01f0 ca09               st bitdly(p2)     ; stor into bit delay

01f2 c416               ldi x'16
01f4 8f00               dly 00            ; short delay

01f6 ba09   rbyte5:     dld bitdly(p2)    ; wait bit delay
01f8 9cfc               jnz rbyte5        ; wait

01fa 19                 sio               ; shift in bit
01fb ba08               dld bitcnt(p2)    ; decrement bit count
01fd 9cef               jnz rbyte4        ; more bits? loop

01ff c215               ld speed(p2)      ; reload delay
0201 ca09               st bitdly(p2)

0203 ba09   rbyte6:     dld bitdly(p2)    ; wait for another bit (stop bit)
0205 9cfc               jnz rbyte6

0207 40                 lde               ; get byte read
0208 3f                 xppc p3           ; return from call
0209 90c6               jmp rbyte         ; loop back to begin

            ; wait for a key press
020b c414   waitky:     ldi l(return)-1   ; set p3 = return helper
020d 33                 xpal p3
020e c400               ldi h(return)
0210 37                 xpah p3

0211 c401               ldi l(hexio1)     ; set p1 = hexio1
0213 31                 xpal p1
0214 c407               ldi h(hexio1)
0216 35                 xpah p1

0217 c4e0               ldi l(stktop)     ; set p2 = stktop
```

```
0219 32                 xpal p2
021a c404               ldi h(stacktop)
021c 36                 xpah p2

021d c108   kyloop:     ld 8(p1)          ; read keyboard (hexio1+8)
021f 94fc               jp kyloop         ; wait for key pressed (bit 7=1)
0221 8f1e               dly x'1e          ; debounce
0223 c108               ld 8(p1)          ; read keyboard again
0225 ca08               st keycod(p2)     ; store key code
0227 d40f               ani x'0f          ; mask out hex data value (if any)
0229 ca09               st keyhex(p2)     ; store data value
022b 01                 xae               ; save into e

022c c108   kyrel:      ld 8(p1)          ; read keyboard
022e 9402               jp ky1            ; wait for key released
0230 90fa               jmp kyrel         ; loop

0232 8f1e   ky1:        dly x'1e          ; debounce

0234 c41f               ldi l(hextbl)-1   ; load 7seg hex table into p1
0236 31                 xpal p1           ; note: offset -1
0237 c401               ldi h(hextbl)
0239 35                 xpah p1

023a c180               ld x'80(p1)       ; index via E
023c ca07               st key7sg(p2)     ; store into key7sg
023e 3f                 xppc p3           ; return

            ; get and display an address (put into addrh/l
023f c406   getadr:     ldi l(hexio6)     set p1 = hexio+6
0241 31                 xpal p1
0242 c407               ldi h(hexio6)
0244 35                 xpah p1

0245 c4e7               ldi l(stktop)+7   set p2 = stacktop+7
0247 32                 xpal p2
0248 c40f               ldi h(stktop+7)
024a 36                 xpah p2

024b c404               ldi x'04          ; count for nibbles
024d caf9               st -7(p2)         ; store in dta

024f c455   adloop:     ldi l(call)-1     ; set p3 = call helper
0251 33                 xpal p3
0252 c400               ldi h(call)
0254 37                 xpah p3

0255 c40a               ldi l(waitky)-1   ; prepare waitkey
0257 cba8               st call-1-calll(p3)     ; via p3
0259 c402               ldi h(waitky)
025b cba7               st call-1-callh(p3)
025d 3f                 xppc p3           ; wait key

025e c4e0               ldi l(stktop)     ; set p3 = stacktop
0260 33                 xpal p3
0261 c40f               ldi h(stktop)
0263 37                 xpah p3

0264 c307               ld key7sg(p3)     ; get 7seg val of key pressed
0266 cdff               st @-1(p1)        ; store into display
0268 c400               ldi x'00          ; clear other fields of display
026a c9ff               st -1(p1)
```

```
026c c9fe              st -2(p1)
026e c9fd              st -3(p1)
0270 c9fc              st -4(p1)
0272 c9fb              st -5(p1)

0274 c309              ld keyhex(p3)     ; get hexcode of key
0276 ceff              st @-1(p2)        ; store into addrl
0278 bb00              dld dta(p3)       ; decrement digit count
027a 9cd3              jnz adloop        ; if not done yet loop

027c c480              ldi x'80          ; set dots in display for data value
027e c9ff              st -1(p1)
0280 c9fe              st -2(p1)

0282 c306              ld digit1(p3)     ; get first digit
0284 1e                rr                ; rotate 4 bits right into H nibble
0285 1e                rr
0286 1e                rr
0287 1e                rr
0288 01                xae               ; into E
0289 c305              ld digit2(p3)     ; get next digit
028b 58                ore               ; merge
028c cb02              st addrh(p3)      ; store into high address
028e c304              ld digit3(p3)     ; get next digit
0290 1e                rr                ; rotate 4 bits right into H nibble
0291 1e                rr
0292 1e                rr
0293 1e                rr
0294 01                xae               ; into E
0295 c303              ld digit4(p3)     ; get last digit
0297 58                ore               ; merge
0298 cb01              st addrl(p3)      ; store into low address

           ; return from subroutine
029a c40037c4124333f
           goret:      js p3,return      ; return from subroutine

02a1 c4e0  disply:     ldi l(stktop)     ; set p3 = stacktop
02a3 33                xpal p3
02a4 c40f              ldi h(stktop)
02a6 37                xpah p3

02a7 c4e0              ldi l(stktop)     ; set p2 = stacktop
02a9 32                xpal p2
02aa c40f              ldi h(stktop)
02ac 36                xpah p2

02ad c4e3              ldi l(stktop+digit4)    ; set p1 = digit4
02af 31                xpal p1
02b0 c40f              ldi h(stktop+digit4)
02b2 35                xpah p1

02b3 c403              ldi x'03          ; number of bytes to unpack
02b5 cb0f              st outb(p3)       ; store in temp count

02b7 c200  unpack:     ld dta(p2)        ; get data byte
02b9 d40f              ani x'0f          ; mask out low nibble
02bb cd01              st @1(p1)         ; store into digit4 and following
02bd c601              ld @1(p2)         ; get databyte again, and point to next
01bf 1c                sr                ; shift high nibble into low
02c0 1c                sr
02c1 1c                sr
```

```
02c2 1c                 sr
02c3 cd01               st @1(p1)         ; store into digit3 and following

02c5 bb0f               dld outb(p3)      ; decrement byte count
02c7 9cee               jnz unpack        ; loop until all bytes unpacked

02c9 c41f               ldi l(hextbl)-1   set p1 = hextable
02cb 31                 xpal p1
02cc c401               ldi h(hextbl)
02ce 35                 xpah p1

02cf c406               ldi x'06          ; store nibbles to convert
02d1 cb0f               st outb(p3)

02d3 c601   cvthex:     ld @1(p2)         ; get nibble
02d5 01                 xae               ; into E
02d6 c180               ld x'80(p1)       ; index via E: get 7seg code
02d8 ca05               st 5(p2)          ; store at stacktop+8+N
02da bb0f               dld outb(p3)      ; decrement loop count
02dc 9cf5               jnz cvthex        ; loop until done

02de c400               ldi l(hexio)      ; set p1 = hexio
02e0 31                 xpal p1
02e1 c407               ldi h(hexio)
02e3 35                 xpah p1

02e4 c406               ldi x'06          ; 6 bytes to display
02e6 cb0f               st outb(p3)
dsply:
02e8 c601   dsloop:     ld @1(p2)         ; get 7 seg byte
02ea cd01               st @1(p1)         ; store into display
02ec bb0f               dld outb(p3)      ; decrement count
02ee 9cf8               jnz dsloop        ; loop for 6 bytes

02f0 90a8               jmp goret         ; return from sub

            ; CAS entry point
02f2 c439   docas:      ldi x'39          ; p1 is hexio1
02f4 c906               st 6(p1)          ; 'C' in display
02f6 c45f               ldi x'5f          ; 'A' in display
02f8 c905               st 5(p1)

02fa 01                 xae               ; set E = 5f (bit 0='1')
02fb 19                 sio               ; shift out a '1'
02fc c4ff               ldi x'ff          ; set 255
02fe ca10               st adflg(p2)      ; into adflg
0300 c40037c455333f     js p3,call        ; do call helper
                                          ; callh/l is still set to waitkey

0307 c45f               ldi x'5f          ; 'A' in display
0309 c900               st 0(p1)
030b c45e               ldi x'5e          ; 'd' in display
030d c9ff               st -1(p1)

030f c208               ld keycod(p2)     ; get keycode
0311 e4e0               xri ky_mod        ; is modify key?
0313 9c1e               jnz cas1          ; no skip

0315 c454               ldi x'54          ; 'm' in display
0317 c900               st 0(p1)
0319 c45c               ldi x'5c          ; 'o' in display
031b c9ff               st -1(p1)
```

```
031d c43e              ldi l(getadr)-1   ; call getadr
031f ca1d              st calll(p2)
0321 3f                xppc p3

0322 c201              ld addrl(p2)      ; low value of argument
0324 ca15              st speed(p2)      ; store into speed

0326 c40a              ldi l(waitky)-1   ; call waitky
0328 ca1d              st calll(p2)
032a 3f                xppc p3

032b c45f              ldi x'5f          ; 'A' in display
032d c900              st 0(p1)
032f c45e              ldi x'5e          ; 'd' in display
0331 c9ff              st -1(p1)

0333 c208   cas1:      ld keycod(p2)     ; get keycode
0335 e480              xri ky_up         ; is UP key?
0337 982c              jz casrd2         ; yes goto casrd

            ; this code is used for both write and read
            ; get a start and end address
0339 c43e   caswr:     ldi l(getadr)-1   ; call getadr (start address)
033b ca1d              st calll(p2)
033d 3f                xppc p3

033e c201              ld addrl(p2)      ; move address into cash/l
0340 ca0b              st casl(p2)
0342 c202              ld addrh(p2)
0344 ca0c              st cash(p2)

0346 3f                xppc p3           ; read end address (remains in addrh/l)

0347 c40a              ldi l(waitky)-1   ; call waitkey
0349 ca1d              st calll(p2)
034b 3f                xppc p3

034c c208              ld keycod(p2)     ; get keycode
034e e480              xri ky_up         ; is UP key?
0350 9c04              jnz caswr0        ; no skip

0352 ca10              st adflg(p2)      ; set flag = 0 (has alternative load address)
0354 900f              jmp casrd2

0356 e480   caswr0:    xri ky_up         ; restore keycode
0358 e490              xri ky_dn         ; is DOWN key?
035a 9802              jz gocasw         ; yes goto cas write
035c 9050              jmp gores         ; no goto reset

035e c40437c4e3333f    gocasw: js p3,caswr     ; goto caswr

0365 c41c   casrd2:    ldi x'1c          ; 'u' in display
0367 c900              st 0(p1)
0369 c473              ldi x'73          ; 'P' in display
036b c9ff              st -1(p1)

036d c4d0              ldi l(rbyte)-1    ; p3 = rbyte routine
036f 33                xpal p3
0370 c401              ldi h(rbyte)
0372 37                xpah p3
```

```
0373 c210               ld adflg(p2)      ; is flag 0? (has alt load addr)
0375 980e               jz skipad         ; yes skip address

0377 3f                 xppc p3           ; get byte
0378 ca0c               st cash(p2)       ; store start addr
0379 3f                 xppc p3
037a ca0b               st casl(p2)

037c 3f                 xppc p3           ; get byte
037d ca02               st addrh(p2)      ; store end addr
0380 3f                 xppc p3
0381 ca01               st addrl(p2)

0383 9004               jmp casrd3        ; goto data reader

0385 3f     skipad:     xppc p3           ; skip 4 bytes
0386 3f                 xppc p3           ; using alternative load address
0387 3f                 xppc p3
0388 3f                 xppc p3

0389 c420   casrd3:     ldi x'20          ; initialize block length
038b ca05               st blkcnt(p2)
038d c400               ldi x'00          ; clear checksum
038f ca06               st cksum(p2)
0391 02                 ccl               ; clear carry for checksum add

0392 c20b   casrd5:     ld casl(p2)       ; load start address into p1
0394 31                 xpal p1
0395 c20c               ld cash(p2)
0397 35                 xpah p1

0398 3f                 xppc p3           ; get byte
0399 c900               st 0(p1)          ; store via p1
039b f206               add cksum(p2)     ; add to checksum
039d ca06               st cksum(p2)      ; store new checksum

039f 35                 xpah p1           ; get pointer H
03a0 e202               xor addrh(p2)     ; compare with end
03a2 9c11               jnz casrd4        ; not at end
03a4 31                 xpal p1           ; get pointer L
03a5 e201               xor addrl(p2)     ; compare with end
03a7 9c0c               jnz casrd4        ; not at end

03a9 3f                 xppc p3           ; get final checksum
03aa e206               xor cksum(p2)     ; compare with checksum
03ac 9c21               jnz caserr        ; not same, then error

03ae c40f37c4ff333f
            gores:      js p3,cold ; cold reset

03b5 06     casrd4:     csa               ; save status
03b6 01                 xae               ; into E

03b7 02                 ccl               ; clear carry
03b8 c20b               ld casl(p2)       ; add 1 to start address
03ba f401               adi 1
03bc ca0b               st casl(p2)
03be c20c               ld cash(p2)
03c0 f400               adi 0
03c2 ca0c               st cash(p2)

03c4 40                 lde               ; get status
```

```
03c5 07                 cas               ; restore it

03c6 ba05               dld blkcnt(p2)    ; decrement block count
03c8 9cc8               jnz casrd5        ; not end of block?

03ca 3f                 xppc p3           ; get block checksum
03cb e206               xor cksum(p2)     ; compare with calculated
03cd 98ba               jz casrd3         ; same, loop

            ; error occurred
03cf c401   caserr:     ldi l(hexio1)     ; point to display
03d1 31                 xpal p1
03d2 c407               ldi h(hexio1)
03d4 35                 xpah p1

03d5 c400               ldi x'00          ; ' Error' in display
03d7 c904               st 4(p1)
03d9 c479               ldi x'79
03db c903               st 3(p1)
03dd c450               ldi x'50
03df c902               st 2(p1)
03e1 c901               st 1(p1)
03e3 c9ff               st -1(p1)
03e5 c45c               ldi x'5c
03e7 c900               st 0(p1)
03e9 90fe   caser1:     jmp caser1        ; endless loop

            ; SUB entry point
03eb c46d   dosub:      ldi x'6d          ; p1 is hexio1
03ed c906               st 6(p1)          ; 's' in display
03ef c476               ldi x'76          ; 'h' in display
03f1 c905               st 5(p1)

03f3 c43e               ldi l(getadr)-1   ; call getadr
03f5 ca1d               st calll-stacktop(p2)
03f7 c40037c455333f     js p3,call

03fe c440               ldi 40            ; '-' in display
0400 c900               st 0(p1)
0402 c400               ldi x'00          ; ' ' in display
0404 c9ff               st -1(p1)

0406 c906               st 6(p1)          ; clear 'sh' in display
0408 c905               st 5(p1)

040a c202               ld addrh(p2)      ; copy address into subtra buffer
040c ca14               st subh(p2)
040e c201               ld addrl(p2)
0410 ca13               st subl(p2)

0412 3f                 xppc p3           ; get another address

0413 03                 scl               ; set carry for subtraction
0414 c213               ld subl(p2)       ; get 1st op l
0416 fa01               cad addrl(p2)     ; subtract
0418 ca01               st addrl(p2)      ; store l result
041a c214               ld subh(p2)       ; get 1st op h
041c fa02               cad addrh(p2)     ; subtract
041e ca02               st addrh(p2)      ; store h result

0420 c40a               ldi l(waitky)-1   ; call wait key
0422 ca1d               st calll(p2)
```

```
0424 3f                 xppc p3

0425 c4a0               ldi l(disply)-1   ; call display
0427 ca1d               st calll(p2)
0429 3f                 xppc p3

042a c400               ldi x'00          ; clear data field in display
042c c9ff               st -1(p1)
042e c900               st 0(p1)
0430 c448               ldi x'48          ; put '=' in display
0432 c905               st 5(p1)
0434 90fe   sub1:       jmp sub1          ; endless loop

            ; CPU entry point
0436 c439   docpu:      ldi x'39          ; p1 = hexio1
0438 c906               st 6(p1)          ; 'C' in display
043a c473               ldi x'73          ; 'P' in display
043c c905               st 5(p1)

043e c43e               ldi l(getadr)-1   ; call getadr
0440 ca1d               st calll(p2)
0442 c40037c455333f     js p3,call

0449 c201               ld addrl(p2)      ; copy start address
044b ca0e               st cpul(p2)
044d c202               ld addrh(p2)
044f ca0d               st cpuh(p2)

0451 3f                 xppc p3           ; call getadr

0452 c201               ld addrl(p2)      ; breakpoint address
0454 31                 xpal p1
0455 c202               ld addrh(p2)
0457 35                 xpah p1           ; into p1

0458 c43f               ldi x'3f          ; save a XPPC 3 at breakpoint
045a c900               st 0(p1)

045c c471               ldi l(retcpu)-1   ; set return point
045e ca1d               st calll(p2)
0460 c404               ldi h(retcpu)
0462 ca1c               st callh(p2)

0464 c20e               ld cpul(p2)       ; get start address
0466 01                 xae               ; save
0467 c20d               ld cpuh(p2)
0469 36                 xpah p2           ; into P2
046a 40                 lde
046b 32                 xpal p2

046c c6ff               ld @-1(p2)        ; point to position before
046e c455               ldi l(call)-1     ; load P3 with call helper
0470 33                 xpal p3           ; note:  P3H is 0 from last call

0471 3e                 xppc p2           ; goto start of program

            ; will return here on breakpoint
0472 c4e0   retcpu:     ldi l(stktop)     ; load stacktop into P2
0474 32                 xpal p2
0475 c40f               ldi h(stktop)
0477 36                 xpah p2
```

```
0478 c4d5              ldi l(stktop)-x'0b     ; point to last stack frame
047a ca1f              st spl(p2)        ; store as new SP

047c c40a              ldi l(waitky)-1   ; call wait key
047e ca1d              st calll(p2)
0480 c402              ldi h(waitky)
0482 ca1c              st callh(p2)
0484 c40037c455333f    js p3,call

048b c208              ld keycod(p2)     ; get keycode
048d 01                xae               ; into E

048e c4a0              ldi l(disply)-1   ; preset addr of disply
0490 ca1d              st calll(p2)

0492 c401              ldi l(hexio1)     ; point to display
0494 31                xpal p1
0495 c407              ldi h(hexio1)
0497 35                xpah p1

0498 40                lde               ; get keycode
0499 e4fa              xri ky_a          ; is 'A' key?
049b 9816              jz cpua           ; yes display A

049d 40                lde               ; get keycode
049e e4fe              xri ky_e          ; is 'E' key?
04a0 9815              jz cpue           ; yes display E

04a2 40                lde               ; get keycode
04a3 e4f5              xri ky_s          is '5'?
04a5 9814              jz cpus           ; yes display status

04a7 40                lde               ; get keycode
04a8 e4f1              xri ky_1          ; is '1' ?
04aa 9813              jz cpu1           ; yes display P1

04ac 40                lde               ; get keycode
04ad e4f2              xri ky_2          ; is '2'?
04af 9815              jz cpu2           ; yes display P2

04b1 90bf   cploop:    jmp retcpu        ; loop

04b3 c2ff   cpua:      ld -1(p2)         ; load A from stackframe
04b5 901c              jmp cpush1

04b7 c2fe   cpue:      ld -2(p2)         ; load E from stackframe
04b9 9018              jmp cpush1

04bb c2fd   cpus:      ld -3(p2)         ; load status from stackframe
04bd 9014              jmp cpush1

04bf c2fc   cpu1:      ld -4(p2)         ; load P1 from stackframe
04c1 01                xae
04c2 c2fb              ld -5(p2)
04c4 9005              jmp cpshow

04c6 c2fa   cpu2:      ld -6(p2)         ; load P2 from stackframe
04c8 01                xae
04c9 c2f9              ld -7(p2)

04cb ca02   cpshow:    st addrh(p2)      ; store 16 bit data
04cd 40                lde
```

```
04ce ca01               st addrl(p2)
04d0 3f                 xppc p3           ; call disply
04d1 9009               jmp cpuclr

04d3 ca01   cpush1:     st addrl(p2)      ; store 8 bit data
04d5 3f                 xppc p3           ; call disply
04d6 c400               ldi x'00          ; clear higher display nibbles
04d8 c903               st 3(p1)
04da c904               st 4(p1)

04dc c400   cpuclr:     ldi x'00          ; clear data field
04de c900               st 0(p1)
04e0 c9ff               st -1(p1)
04e2 90cd               jmp cploop        ; goto loop

            ; CAS Write entry point
04e4 c45e   caswr:      ldi x'5e          ; p1 is hexio1
04e6 c900               st 0(p1)          ; 'd' in display
04e8 c45c               ldi x'5c          ; 'o' in display
04ea c9ff               st -1(p1)

04ec c20b               ld casl(p2)
04ee 31                 xpal p1
04ef c20c               ld cash(p2)
04f1 35                 xpah p1           ; start address into P1

04f2 c4d7               ldi l(wbyte)-1    ; preset wbyte routine
04f4 33                 xpal p3
04f5 c405               ldi h(wbyte)
04f7 37                 xpah p3

04f8 c20c               ld cash(p2)       ; write start address
04fa 3f                 xppc p3           ; to tape
04fb c20b               ld casl(p2)
04fd 3f                 xppc p3

04fe c202               ld addrh(p2)      ; write end address
0500 3f                 xppc p3           ; to tape
0501 c201               ld addrl(p2)
0503 3f                 xppc p3

0504 c420   caswr1:     ldi x'20          ; set block count
0506 ca05               st blkcnt(p2)     ; into temporary
0508 c400               ldi x'00          ; clear checksum
050a ca06               st cksum(p2)
050c 02                 ccl               ; clear carry for checksum add

caswr2:
050d c100   caswr2:     ld 0(p1)          ; get first byte
050f 01                 xae               ; into E
0510 c206               ld cksum(p2)      ; get checksum
0512 70                 ade               ; add
0513 ca06               st cksum(p2)      ; save checksum

0515 40                 lde               ; get byte
0516 3f                 xppc p3           ; write byte

0517 35                 xpah p1           ; get H start address
0518 e202               xor addrh(p2)     ; compare with H end
051a 01                 xae               ; into E
051b 40                 lde               ; restore
051c e202               xor addrh(p2)     ; restore into P1
```

```
051e 35                 xpah p1

051f 40                 lde             ; get compare result
0520 9c08               jnz caswr3      ; not same, skip

0522 31                 xpal p1         ; get L start
0523 e201               xor addrl(p2)   ; compare with L end
0525 9819               jz caswr4       ; end reached, skip
0527 e201               xor addrl(p2)   ; restore P1
0529 31                 xpal p1

052a 06     caswr3:     csa             ; save status
052b 01                 xae             ; into E
052c 02                 ccl             ; clear carry
052d 31                 xpal p1         ; increment P1
052f f401               adi 1
0530 31                 xpal p1
0531 35                 xpah p1
0532 f400               adi 0
0534 35                 xpah p1
0535 40                 lde             ; restore status
0536 07                 cas

0537 ba05               dld blkcnt(p2)  ; decrement block count
0539 9cd2               jnz caswr2      ; no end of block, loop

053b c206               ld cksum(p2)    ; get checksum
053d 3f                 xppc p3         ; write byte
053e 90c4               jmp caswr1      ; loop

0540 c206   caswr4:     ld cksum(p2)    ; last block, write chksum
0542 3f                 xppc p3

0543 c40f37c4ff333f
            caswr5:     js p3,cold      ; goto reset

            ; BLK entry point
054a c47c   doblk:      ldi x'7c        ; p1 is hexio1
054c c906               st 6(p1)        ; 'B' in display
054e c438               ldi x'38        ; 'L' in display
0550 c905               st 5(p1)

0552 c43e               ldi l(getadr)-1 ; call getadr
0554 ca1d               st calll(p2)
0556 c40037c455333f     js p3,call

055d c201               ld addrl(p2)
055f ca10               st blbegl(p2)
0561 c202               ld addrh(p2)
0563 ca0f               st blbegh(p2)   ; save start address

0565 3f                 xppc p3         ; call getadr
0566 c201               ld addrl(p2)
0568 ca0e               st blendl(p2)
056a c202               ld addrh(p2)
056c ca0d               st blendh(p2)   ; save end address

056e 3f                 xppc p3         ; call getadr

056f c40a               ldi l(waitky)-1 ; call wait key
0571 ca1d               st calll(p2)
0573 3f                 xppc p3
```

```
0574 03                 scl               ; set carry for subtraction
0575 c201               ld addrl(p2)      ; target
0577 fa10               cad blbegl(p2)    ; - start
0579 ca0c               st bldifl(p2)     ; store delta
057b c202               ld addrh(p2)
057d fa0f               cad blbegh(p2)
057f ca0b               st bldifh(p2)
0581 9429               jp blkpos         ; if positive skip

0583 c210               ld blbegl(p2)
0585 31                 xpal p1
0586 c20f               ld blbegh(p2)
0588 35                 xpah p1           ; start address into P1

0589 c201               ld addrl(p2)      ; target into p3
058b 33                 xpal p3
058c c202               ld addrh(p2)
058e 37                 xpah p3

058f c501   blkn1:      ld @1(p1)         ; get byte
0591 cf01               st @1(p3)         ; copy to target
0593 c5ff               ld @-1(p1)        ; decrement PTR
0595 31                 xpal p1           ; compare P1 with end address
0596 e20e               xor blendl(p2)
0598 01                 xae
0599 40                 lde
059a e20e               xor blendl(p2)
059c 31                 xpal p1           ; restore P1L
059d 40                 lde
059e 9c08               jnz blkn3         ; not yet finished
05a0 35                 xpah p1
05a1 e20d               xor blendh(p2)

05a3 989e   blkn2:      jz caswr5         ; finished, cold reset
05a5 e20d               xor blendh(p2)    ; restore P1H again
05a7 35                 xpah p1

05a8 c501   blkn3:      ld @1(p1)         ; increment P1
05aa 90e3               jmp blkn1         ; loop next byte

05ac c20e   blkpos:     ld blendl(p2)
05ae 31                 xpal p1
05af c20d               ld blendh(p2)
05b1 35                 xpah p1           ; load end address into P1
05b2 c501               ld @1(p1)         ; increment P1

05b4 03                 scl               ; add 1 + delta
05b5 c20e               ld blendl(p2)
05b7 f20c               add bldifl(p2)
05b9 33                 xpal p3           ; set p3 = target address
05ba c20d               ld blendh(p2)
05bc f20b               add bldifh(p2)
05be 37                 xpah p3

05bf c5ff   blkp1:      ld @-1(p1)        ; load source byte
05c1 cfff               st @-1(p3)        ; store at target pos

05c3 31                 xpal p1           ; copy downwards
05c4 e210               xor blbegl(p2)    ; compare with blbeg
05c6 01                 xae
05c7 40                 lde
```

```
05c8 e210              xor blbegl(p2)
05ca 31                xpal p1
05cb 40                lde
05cc 9cf1              jnz blkp1         ; not yet done, loop
05ce 35                xpah p1
05cf e20f              xor blbegh(p2)
05d1 98d0              jz blkn2          ; done now, goto cold start
05d3 e20f              xor blbegh(p2)    ; restore P2H
05d5 35                xpah p1
05d6 90e7              jmp blkp1         ; loop next byte

           ; write a byte serially to tape
05d8 ca07  wbyte:      st bytesv(p2)     ; save byte
05da c40b              ldi x'0b          ; 1 startbit, 8 data, 2 stop bits
05dc ca08              st bitcnt(p2)     ; save bitcount
05de c400              ldi x'00          ; send startbit
05e0 01                xae
05e1 19                sio               ; shift out
05e2 01                xae
05e3 ba20              dld 20(p2)        ; waste time
05e5 c207              ld bytesv(p2)     ; get byte
05e7 01                xae               ; into E

05e8 c40b  wbyte1:     ldi x'0b          ; delay
05ea 8f00              dly 00
05ec c215              ld speed(p2)      ; get bit speed
05ee ca09              st bitdly(p2)     ; stor into temp

05f0 ba09  wbyte2:     dld bitdly(p2)    ; wait
05f2 9cfc              jnz wbyte2

05f4 19                sio               ; send out bit
05f5 40                lde               ; get data
05f6 dc80              ori x'80          ; set top bit (will be stop bit then)
05f8 01                xae               ; back into E

05f9 ba08              dld bitcnt(p2)    ; decrement bit count
05fb 9ceb              jnz wbyte1        ; not yet done, loop

05fd 3f                xppc p3           ; return from call
05fe 90d8              jmp wbyte         ; reenter loop to routine
```