```
;title  ' H19 Driver ver 2.2 '
;*********************************************************
;*      last update: June 3, 1983 / Friday       *
;*      programmer : Les Bird                    *
;*      company    : CompTec Software Dept.      *
;*      language   : 8080 Assembly               *
;*      O S        : CP/M ver 2.0                *
;*********************************************************
;
; H19 escape codes
;
esc     equ     1bh     ; escape
hcuh    equ     'H'     ; home cursor
hcuf    equ     'C'     ; cursor right
hcub    equ     'D'     ; cursor left
hcud    equ     'B'     ; cursor down
hcuu    equ     'A'     ; cursor up
hri     equ     'I'     ; reverse index
hcpr    equ     'n'     ; cursor position report
hscp    equ     'j'     ; save cursor position
hrcp    equ     'k'     ; return to saved
hdca    equ     'Y'     ; direct addressing
hcd     equ     'E'     ; clear screen
hbd     equ     'b'     ; erase beginning
heop    equ     'J'     ; end of page
hel     equ     'l'     ; entire line
hebl    equ     'o'     ; beginning line
heol    equ     'K'     ; end of line
hil     equ     'L'     ; insert line
hdl     equ     'M'     ; delete line
hdch    equ     'N'     ; delete character
heim    equ     '@'     ; enter insert mode
herm    equ     'O'     ; exit insert mode
hram    equ     'z'     ; reset terminal
hsm     equ     'x'     ; set mode
hrm     equ     'y'     ; reset mode
herv    equ     'p'     ; reverse video
hxrv    equ     'q'     ; exit reverse video
hegm    equ     'F'     ; enter graphics mode
hxgm    equ     'G'     ; exit graphics mode
hf1     equ     'S'     ; f1
hf2     equ     'T'     ; f2
hf3     equ     'U'     ; f3
hf4     equ     'V'     ; f4
hf5     equ     'W'     ; f5
hf7     equ     'P'     ; blue
hf8     equ     'Q'     ; red
hf9     equ     'R'     ; white
offset  equ     1fh
;
; bdos entry points
;
reboot: equ     0000    ; reboot system
bdos    equ     5
orgin   equ     100h
direct  equ     6
dinput  equ     0ffh
pstring equ     9
clock:  equ     0bh
cr      equ     0dh
```

```
lf      equ     0ah
bs      equ     8
ap      equ     27h
bel     equ     7
callop  equ     0cdh    ; call op-code.
jmpop   equ     0c3h    ; jmp op-code.
;
        org     orgin
start:  lxi     h,0
        dad     sp
        shld    oldstack
        lxi     sp,stack
        jmp     begin
;
;       JUMP VECTORS
;
        jmp     cls     ; clear screen
        jmp     zsmem   ; zero memory map
        jmp     ceol    ; clear end of line
        jmp     crlf    ; output carriage and linefeed
        jmp     insertl ; insert line
        jmp     deletel ; delete line
        jmp     insertc ; enter insert char mode
        jmp     deletec ; delete character
        jmp     xinsertc; exit insert char mode
        jmp     creport ; cursor position report
        jmp     cup     ; cursor up
        jmp     decrl   ; decrement cursor Y
        jmp     cleft   ; cursor left
        jmp     decrh   ; decrement cursor X
        jmp     cright  ; cursor right
        jmp     incrh   ; increment cursor X
        jmp     cdown   ; cursor down
        jmp     incrl   ; increment cursor Y
        jmp     home    ; cursor home
        jmp     process ; process escape sequence
        jmp     hset    ; Heath set mode
        jmp     clr25   ; clear 25th line
        jmp     hrset   ; Heath reset mode
        jmp     output  ; output a char
        jmp     input   ; check for input
        jmp     wait    ; wait for character
        jmp     waitupr ; wait for character and convert to uppercase
        jmp     bentry  ; BDOS entry point
        jmp     savall  ; save all registers
        jmp     retall  ; retrieve all registers
        jmp     adj     ; adjust cursor position
        jmp     update  ; update memmap
        jmp     rsmem   ; read memmap
        jmp     ssmem   ; store char in memmap
        jmp     graphix ; enter graphics mode
        jmp     xgraphix; exit graphics mode
        jmp     reverse ; enter reverse video mode
        jmp     xreverse; exit reverse video mode
        jmp     cursoff ; cursor off
        jmp     curson  ; cursor on
        jmp     cursl   ; line cursor
        jmp     cursb   ; block cursor
        jmp     savecurs; save cursor position
        jmp     retcurs ; put cursor at saved position
        jmp     boot    ; reboot CP/M
        jmp     restore ; same as above
        jmp     grout   ; output single graphic character
        jmp     spmsg   ; print message from stack
        jmp     show    ; print message from HL
        jmp     delay   ; time delay
        jmp     hldiv   ; H/2, L/2
        jmp     vector  ; draw line
        jmp     box     ; draw box
        jmp     seeline ; checks status of line
        jmp     sound   ; creates sound through H8 speaker
;
; CLEAR DISPLAY
;
cls:    push    h
        lxi     h,101h
        shld    curco
        call    adj     ; incase on 25th line
        mvi     a,hcd
        call    process
        lxi     h,memmap
        shld    scrpnt
        pop     h
zsmem:  push    h
        lxi     h,memmap
        lxi     d,2001
zsmem1: mvi     m,0
        inx     h
        dcx     d
        mov     a,e
        ora     d
        jnz     zsmem1
        pop     h
        ret
;
ceol:   push    h
        mvi     a,heol
        call    process
        lhld    curco
        xchg
        lhld    scrpnt
eolloop:mvi     m,0
        inx     h
        dcr     d
        jnz     eolloop
        pop     h
        ret
crlf:   mvi     a,cr
        call    output
        mvi     a,lf
        call    output
        push    h
        lhld    curco
        mvi     h,1
        inr     l
        shld    curco
        pop     h
        ret
;
insertl:
        mvi     a,hil
        jmp     process
;
deletel:
        mvi     a,hdl
        jmp     process
;
insertc:
        mvi     a,heim
        jmp     process
;
deletec:
        mvi     a,hdch
        jmp     process
;
xinsertc:
        mvi     a,herm
        jmp     process
;
creport:                ; cursor position report
        call    savall  ; save all registers
        mvi     a,hcpr  ;
        call    process ;
creport1:
        call    wait    ; get ESC
        cpi     esc     ; check it
        jnz     creport1; loop until ESC
creport2:
        call    wait    ; get "Y"
        cpi     'Y'     ; check it
        jnz     creport2;
        call    wait    ; get y
        sui     31      ; subtract offset
        sta     curco+1 ; cursor y storage
        call    wait    ; get x
        sui     31      ; subtract offset
        sta     curco   ; cursor x storage
        call    retall  ; retrieve all registers
        ret
;
cup:    mvi     a,hcuu
        call    process
decrl:  call    savall
        lhld    curco
        dcr     l
        shld    curco
        lhld    scrpnt
        lxi     d,-80
        dad     d
        shld    scrpnt
        call    retall
        ret
;
cleft:  mvi     a,hcub
        call    process
decrh:  call    savall
        lhld    curco
        dcr     h
        shld    curco
        lhld    scrpnt
        dcx     h
        shld    scrpnt
        call    retall
        ret
;
cright: mvi     a,hcuf
        call    process
incrh:  call    savall
        lhld    curco
        inr     h
        shld    curco
        lhld    scrpnt
        inx     h
        shld    scrpnt
        call    retall
        ret
;
cdown:  mvi     a,hcud
        call    process
incrl:  call    savall
        lhld    curco
        inr     l
        shld    curco
        lhld    scrpnt
        lxi     d,80
        dad     d
        shld    scrpnt
        call    retall
        ret
;
home:   push    h
        mvi     a,hcuh
        call    process
        lxi     h,101h
        shld    curco
        lxi     h,memmap
        shld    scrpnt
        pop     h
        ret
;
process:
        push    psw
        mvi     a,esc
        call    output
        pop     psw
        jmp     output
hset:   push    psw
        mvi     a,hsm
hsetm:  call    process
        pop     psw
        call    output
        ret
;
clr25:  mvi     a,'1'
        call    hrset
        mvi     a,'1'
        jmp     hset
;
hrset:  push    psw
        mvi     a,hrm
        jmp     hsetm
;
output: mov     e,a
        mvi     c,direct
        jmp     bentry
output1:lhld    retvec
        pchl
;
input:  mvi     e,dinput
        mvi     c,direct
        jmp     bentry
;
wait:   call    input
        ora     a
        jz      wait
        ret
;
waitupr:
        call    wait
        cpi     'a'
        rc
        ani     5fh     ; make uppercase
        ret
;
bentry: push    b
        push    d
        push    h
        call    bdos
        pop     h
        pop     d
        pop     b
        ret
;
savall: xthl
        push    d
        push    b
        push    psw
        push    h
        ret
;
retall: pop     h
        pop     psw
        pop     b
        pop     d
        xthl
        ret
;
adj:    push    h
        mvi     a,hdca
        call    process
        mov     a,l
        adi     offset
        call    output
        mov     a,h
        adi     offset
        call    output
        pop     h
adjmem: push    h
        mov     b,h
        mov     c,l
        lxi     h,memmap
        lxi     d,50h
        dcr     c
        jz      adj1
adjloop:dad     d
        dcr     c
        jnz     adjloop
adj1:   mov     e,b
        mvi     d,0
        dad     d
        shld    scrpnt
        pop     h
        ret
;
update: push    h
        lhld    scrpnt
        mov     m,a
        inx     h
        shld    scrpnt
        pop     h
        ret
;
rsmem:  push    h
        call    adjmem  ; 01/18/83
        lhld    scrpnt
        mov     a,m
        pop     h
        ret
;
ssmem:  push    h
        push    psw     ; save character
        call    adjmem  ; 01/18/83
        lhld    scrpnt
        pop     psw     ; get character back
        mov     m,a
        pop     h
        ret
;
graphix:
        mvi     a,hegm
        call    process
        mvi     a,1
        sta     gbit
        ret
;
xgraphix:
        mvi     a,hxgm
        call    process
        mvi     a,0
        sta     gbit
        ret
;
reverse:
        mvi     a,herv
        call    process
        mvi     a,1
        sta     rbit
        ret
;
xreverse:
        mvi     a,hxrv
        call    process
        mvi     a,0
        sta     rbit
        ret
;
cursoff:
        mvi     a,'x'
        call    process
        mvi     a,'5'
        jmp     output
;
curson:
        mvi     a,'y'
        call    process
        mvi     a,'5'
        jmp     output
;
cursl:  mvi     a,hrm
        call    process
        mvi     a,'4'
        jmp     output
;
cursb:  mvi     a,hsm
        call    process
        mvi     a,'4'
        jmp     output
;
savecurs:
        mvi     a,hscp
        jmp     process
;
retcurs:
        mvi     a,hrcp
        jmp     process
;
boot:
restore:
        mvi     a,hram
        call    process
        lhld    oldstack
        sphl            ; set stack
        ret             ; return to CCP
;
grout:  sta     grchar  ; save character
        call    graphix ; graphics on.
        lda     grchar  ; get character
        call    output  ; output it.
        call    xgraphix;
        ret
;
spmsg:  xthl            ; SP=HL  HL=SP
        call    show    ; print message
        xthl            ; SP=HL  HL=SP
        ret             ; finished
;
show:   mov     a,m
        cpi     '@'
        jz      printat
        cpi     '['
        jz      special
        cpi     0
        rz
        inx     h
        call    update
        call    output
        jmp     show
printat:
        inx     h
        mov     c,m
        inx     h
        mov     b,m
        inx     h
        call    savall
        mov     l,c
        mov     h,b
        call    adj
        call    retall
        jmp     show
special:
        inx     h
        mov     a,m
        cpi     'R'
        cz      reverse
        cpi     'r'
        cz      xreverse
        cpi     'G'
        cz      graphix
        cpi     'g'
        cz      xgraphix
        cpi     'S'
        cz      setmodes
        cpi     's'
        cz      rsetmodes
        cpi     'C'
        cz      cls
        cpi     'c'
        cz      ceol
        cpi     ']'
        jnz     special
        inx     h
        jmp     show
setmodes:
        mvi     a,hsm
setentry:
        call    process
        inx     h
        mov     a,m
        jmp     output
rsetmodes:
        mvi     a,hrm
        jmp     setentry
delay:  call    savall
        lhld    time
delay1: dcx     h
        mov     a,h
        ora     l
        jnz     delay1
        call    retall
        ret
```

```
hldiv:  push    psw
        mov     a,h
        rar             ; H/2
        mov     h,a     ; H=H/2
        sbb     a       ; subtract with borrow
        mov     b,a     ; B=0ffh if carry, 0 if not
        mov     a,l
        rar             ; L/2
        mov     l,a     ; L=L/2
        sbb     a       ; get remainder
        mov     c,a     ; C=0ffh if carry
        pop     psw     ;
        ret
vector: lhld    linco   ; point #1
        xchg            ; to DE.
        lhld    linco1  ; point #2 to HL.
        lxi     b,0000h ; push stack empty flag
        push    b       ;
        mov     a,d     ;
        cmp     h       ;
        jc      noxg    ;
        jnz     xg      ;
        mov     a,e     ;
        cmp     l       ;
        jc      noxg    ;
xg:     xchg            ;
noxg:   mov     b,h     ;
        mov     c,l     ;
        lxi     h,inral ;
        mvi     m,3ch   ;
        mov     a,e     ;
        cmp     c       ;
        jc      loop    ;
        mvi     m,0ch   ;
; origin = DE
; endpt  = BC
; midpt  = HL
loop:   mov     a,d     ;
        cmp     b       ;
        jz      eqx     ;
        push    b       ;
        add     b       ;
        rar             ;
        mov     b,a     ;
        inr     a       ;
        mov     h,a     ;
        mov     a,e     ;
        cmp     c       ;
        jz      eqy     ;
neqy:   add     c       ;
        rar             ;
        mov     c,a     ;
inral:  inr     a       ;
eqy:    mov     l,a     ;
        push    h       ;
        jmp     loop    ;
eqx:    mov     a,e     ;
        cmp     c       ;
        jz      eqxy    ;
        push    b       ;
        mov     h,d       ;
        jmp     neqy      ;
vplot:                    ; either (rsmem) or (plot)
eqxy:   call    plot      ;
eqxy1:  pop     d         ;
        mov     a,d       ;
        ora     e         ;
        jz      vexit     ; exit line routine
        pop     b         ;
        jmp     loop      ;
vexit:  mvi     a,0       ; finished flag
        ret               ; return
;
; PLOT routine plots points according
; to the VECTOR routine.
; PLOT pixels  48 X 160.  Where X = 1 to 160 & Y = 1 to 48
;
plot:   xra     a         ; clear carry flag
        mov     a,e       ; get y
        rar               ; divide by 2
        mov     l,a       ; store in L
        jnc     plotop    ; plot top section
        xra     a         ; clear carry
        mov     a,d       ; get x
        rar               ; divide by 2
        mov     h,a       ; store in H
        jnc     plotbl    ; bottom left
        mvi     b,8       ; bottom right
        jmp     plot1     ; plot point
plotbl: mvi     b,4       ; bottom left point
        jmp     plot1     ; plot point
plotop: xra     a         ; carry=0
        mov     a,d       ; get x
        rar               ; divide by 2
        mov     h,a       ; store in H
        jnc     plotl     ; top left
        mvi     b,2       ; top right
        jmp     plot1     ; set point
plotl:  mvi     b,1       ; top left
plot1:  push    b         ; save BC
        call    rsmem     ; read memmap
        pop     b         ; get BC back
        sta     astor     ; save character
        ana     b         ; and with B
        rnz               ; bit already set
        lda     astor     ;
        ora     b         ; OR with B
        call    ssmem     ; update in memmap
        push    psw       ;
        call    adj       ;
        pop     psw       ;
plot2:  cpi     1         ; is it bit 1?
        jz      point1    ; set bit 1
        cpi     2         ; bit 2?
        jz      point2    ; set bit 2
        cpi     3         ; bit 1 & 2?
        jz      point3    ; set bit 1 & 2
        cpi     4         ; bit 4?
        jz      point4    ; set bit 4
        cpi     5         ; bit 4 & 1?
        jz      point5  ; set bit 4 & 1
        cpi     6       ; bit 4 & 2?
        jz      point6  ; set bit 4 & 2
        cpi     7       ; bit 1,2,4?
        jz      point7  ; set bit 1,2,4
        cpi     8       ; bit 8?
        jz      point8  ; set bit 8
        cpi     9       ; bit 8 & 1?
        jz      point9  ; set bit 8,1
        cpi     0ah     ; bit 8 & 2?
        jz      pointa  ; set bit 8,2
        cpi     0bh     ; bit 1,2,8?
        jz      pointb  ; set bit 1,2,8
        cpi     0ch     ; bit 8 & 4?
        jz      pointc  ; set bit 8,4
        cpi     0dh     ; bit 8,4,1?
        jz      pointd  ; set bit 1,4,8
        cpi     0eh     ; bit 8,4,2?
        jz      pointe  ; set bit 2,4,8
pointf: call    reverse ; enter reverse video
        mvi     a,' '   ; bits 1,2,4,8
        call    output  ; output space
        jmp     xreverse; exit reverse video
pointe: call    reverse ; enter reverse video
        mvi     a,'n'   ; bits 2,4,8
        call    grout   ; output graphics
        jmp     xreverse; exit reverse video
pointd: call    reverse ; reverse video mode
        mvi     a,'o'   ; bits 1,4,8
        call    grout   ; output graphic char
        jmp     xreverse; exit reverse video
pointc: call    reverse ; reverse video
        mvi     a,'p'   ;
        call    grout   ;
        jmp     xreverse;
pointb: call    reverse ; reverse video
        mvi     a,'m'   ;
        call    grout   ;
        jmp     xreverse;
pointa: mvi     a,'q'   ; bits 2,8
        jmp     grout   ; output graphic char
point9: jmp     pointd  ; bits 1,8 +4
point8: mvi     a,'l'   ; bit 8
        jmp     grout   ; output graphic char
point7: call    reverse ; reverse video
        mvi     a,'l'   ; bits 1,2,4
        call    grout   ; graphic char
        jmp     xreverse; exit
point6: jmp     pointe  ; bits 2,4
point5: call    reverse ; reverse video
        mvi     a,'q'   ;
        call    grout   ;
        jmp     xreverse; exit
point4: mvi     a,'m'   ; bit 4
        jmp     grout   ; output graphics
point3: mvi     a,'p'   ; bits 1,2
        jmp     grout   ;
point2: mvi     a,'o'   ; bit 2
        jmp     grout   ;
point1: mvi     a,'n'   ; bit 1
```

```
         jmp     grout    ;
box:     lhld    boxco    ;
         shld    linco    ; x1,y1
         xchg             ;
         lhld    boxco1   ;
         mov     l,e      ; make y2=y1
         shld    linco1   ; x1,y1 -> x2,y1
         call    boxplot  ; save all registers
         lhld    boxco1   ; get original x2,y2
         shld    linco1   ;
         mov     d,h      ; make x1=x2
         xchg             ;
         shld    linco    ; x2,y1 -> x2,y2
         call    boxplot  ;
         lhld    boxco    ; get original x1,y1
         xchg             ;
         lhld    boxco1   ;
         mov     e,l      ; make y1=y2
         xchg             ;
         shld    linco    ; x2,y2 -> x1,y2
         call    boxplot  ;
         lhld    boxco    ; get original x1,y1
         shld    linco    ;
         xchg             ;
         lhld    boxco1   ; original x2,y2
         mov     h,d      ; x1,y2 -> x1,y1
         shld    linco1   ;
boxplot:call     savall   ; save all registers
         call    vector   ; draw line
         call    retall   ; retrieve all regs
         ret              ; return for more
;
;        SEELINE - draws imaginary line and
;                  checks line status.
;
seeline:mvi      a,jmpop  ; jump opcode.
         sta     eqxy     ; store in vector.
         lxi     h,rsmem  ; look at memmap.
         shld    eqxy+1   ; store in vector.
seelin1:call     vector   ; check line.
         cpi     0        ; finished checking?
         jz      seelin2  ; exit
         cpi     ' '      ; empty space?
         jz      eqxy1    ; continue checking.
         jmp     eqxy1    ;
seelin2:mvi      a,callop; call opcode.
         sta     eqxy     ; put in vector.
         lxi     h,plot   ; plot line.
         shld    eqxy+1   ; store in vector.
         ret              ; finished checking.
; SOUND -- creates sounds according to data in SNDMEM.
;             Enter : DE = delay rate
;
sound:   call    savall   ; save everything
sound0:  mvi     a,10h    ; speaker on bit.
         out     0f0h     ; output to port 360Q
sound1:  lda     length   ;
         dcr     a
         sta     length
         jnz     sound1   ; loop for good sound.
```

```
        mvi     a,0f4h  ; speaker off bit.
        out     0f0h    ; port 360Q
        lhld    count   ; delay.
        xchg            ; DE = delay from COUNT.
sound2: dcx     d       ; decrement delay.
        mov     a,d     ;
        ora     e       ; test for zero.
        jnz     sound2  ;
        lhld    repeat  ; times to repeat
        mov     a,h
        ora     l
        jz      sound3  ; finished.
        dcx     h
        shld    repeat
        call    input   ; test for input
        cpi     03h     ; CTRL-C
        jz      sound3  ; exit
        jmp     sound0  ; repeat until HL=0000
sound3: jmp     retall  ; return
;
;       16-bit math package from BYTE, May 1981 - vol 6 #5
;
;       EADD    (HL)=(HL)+(DE)
;       ESUB    (HL)=(HL)-(DE)
;       EMULT   (HL)=(HL)*(DE)
;       EDIVMOD (HL)=(HL)/(DE), (DE)=(HL) MOD (DE)
;       ESIGN   SET (S), (Z) FLAG TO REFLECT (HL)-(DE), LEAVING
;               (HL) UNCHANGED.
;       ECMP    SET (S), (Z) FLAGS TO REFLECT (HL), LEAVING (HL)
;               AND (DE) UNCHANGED.
;
;       DECBIN  CONVERT ASCII CHARACTER STRING REPRESENTING A SIGNED
;               DECIMAL INTEGER TO A SIGNED BINARY NUMBER.
;       BINDEC  CONVERT A SIGNED BINARY NUMBER TO AN ASCII STRING
;               REPRESENTING THE SIGNED DECIMAL VALUE OF THE NUMBER.
;
;       MATH PACKAGE EXECUTION TIMES IN MICRO-SECONDS
;
;       ROUTINE         TYPICAL         WORST CASE
;
;       EADD            30              54
;       ESUB            50              74
;       EMULT           370             517
;       EDIVMOD         680             2500
;
overflow        ret     ; Return if error
converr         ret     ; Same here
;
eadd:                   ; 16-bit addition
        mov     a,h     ; test signs
        xra     d       ;
        ani     80h     ;
        dad     d       ; add, without affecting zero flag...
        jnz     esign   ; skip overflow test if signs differ
        rar             ; test for overflow by...
        xra     h       ; ...exclusive or of CY and sign of result
        ral             ;
        cc      overflow; check for arith overflow
;
;       ESIGN
```

```
;
esign:                  ; set (S), (Z) flags to reflect (HL)
        xra     a       ; clear flags
        add     h       ; set flags to reflect HI byte
        rnz             ; return if HI byte is non-0
        add     l       ; else, see if (L) is 0 too...
        rz              ; and if so, return
        xra     a       ; else, force flags to show '+'
        inr     a       ;
        ret             ;
;
;       ESUB
;
esub:                   ; 16-bit subtraction
        push    d       ;
        xchg            ;
        call    comp2   ; form 25 complement of subtrahend...
        call    eadd    ; ... and proceed as in addition
        pop     d       ;
        ret             ;
;
;       ECHS - CHANGE SIGN OF REGISTER (HL)
;
echs:                   ;
        mov     a,h     ;
        sui     80h     ; check for that one nasty case...
        jnz     echsgo  ; ... of (HL) = -32768...
        add     l       ; ... which can't be complemented right
        cz      overflow; ... and when detected, abort
echsgo:                 ;
        call    comp2   ; else, form 25 complement in (HL)
        jmp     esign   ; set flags and return
;
;       25 COMPLEMENT OF (HL)
;
comp2:                  ;
        mov     a,h     ;
        cma             ;
        mov     h,a     ;
        mov     a,l     ;
        cma             ;
        mov     l,a     ;
        inx     h       ;
        ret
;
;       EMULT
;
emult:                  ;
        push    b       ;
        push    d       ;
        call    rsltsign; find result sign, abs val of operands
        xra     a       ;
        add     h       ;
        jz      hlsmall ; branch if (HL) less than 8 bits
        xra     a       ;
        add     d       ; else, other op must be .lt. 8 bits...
        cnz     overflow; ... or overflow would result
        xchg            ;
hlsmall:                ;
        mov     a,l     ; move 8-bit or less multiplier to (A)
```

```
          lxi     h,0      ; initialize partial product
xmloop:                    ;
          stc              ; clear carry
          cmc              ;
          rar              ; rotate multiplier right off end
          jnc     shiftop  ; if bit shifted-out was 0, skip
          dad     d        ; else, add multiplicand to partial prod.
          cc      overflow;  ... while checking for overflow
shiftop:                   ;
          xchg             ;
          dad     h        ;
          cc      overflow;
          xchg             ;
          ora     a        ;
          jnz     xmloop   ;
          pop     d        ;
signrcl:                   ;
          mov     a,h      ;
          rlc              ;
          cc      overflow;
          mov     a,b      ;
          ral              ;
          cc      comp2    ;
          pop     b        ;
          jmp     esign    ;
;
;         COMPUTE SIGN OF RESULT FOR * AND /
;
rsltsign:                  ;
          mov     b,h      ;
          mov     a,h      ;
          ral              ;
          cc      comp2    ;
          xchg             ;
          mov     a,h      ;
          xra     b        ;
          mov     b,a      ;
          mov     a,h      ;
          ral              ;
          jc      comp2    ;
          ret              ;
;
;         DIVIDE (HL) BY (DE)
;
edivmod:                   ;
          push    b        ;
          xra     a        ;
          ora     e        ;
          ora     d        ;
          cz      overflow;
          call    rsltsign;
          mov     a,h      ;
          ora     d        ;
          rlc              ;
          cc      overflow;
          push    b        ;
          mov     c,e      ;
          mov     b,d      ;
          lxi     d,0      ;
          push    d        ;
```

```
        xchg            ;
        lxi     h,1     ;
dbldiv:                 ;
        dad     h       ;
        xchg            ;
        dad     h       ;
        call    cmpbh   ;
        xchg            ;
        jnc     dbldiv  ;
        xchg            ;
halvediv:               ;
        xchg            ;
        call    divby2  ;
        jz      divdone ;
        xchg            ;
        call    divby2  ;
        call    cmpbh   ;
        jm      halvediv;
        mov     a,c     ;
        sub     l       ;
        mov     c,a     ;
        mov     a,b     ;
        sbb     h       ;
        mov     b,a     ;
        xthl            ;
        dad     d       ;
        xthl            ;
        jmp     halvediv;
divdone:                ;
        pop     h       ;
        mov     e,c     ;
        mov     d,b     ;
        pop     b       ;
        jmp     signrcl ;
cmpbh:                  ;
        mov     a,c     ;
        sub     l       ;
        mov     a,b     ;
        sbb     h       ;
        ret             ;
divby2:                 ;
        xra     a       ;
        mov     a,h     ;
        rar             ;
        mov     h,a     ;
        mov     a,l     ;
        rar             ;
        mov     l,a     ;
        ora     h       ;
        ret             ;
;
;       DECBIN - CONVERT ASCII DECIMAL TO BINARY NUMBER
;
decbin:                 ;
        push    b       ;
        mvi     b,0     ;
        lxi     h,0     ;
akloop:                 ;
        ldax    d       ;
        sui     48      ;
```

```
                mov     c,a     ;
                jm      notdigit;
                cpi     10      ;
                jp      notdigit;
                push    d       ;
                lxi     d,10    ;
                call    emult   ;
                mvi     d,0     ;
                mov     e,c     ;
                call    eadd    ;
                pop     d       ;
                inx     d       ;
                mvi     a,1     ;
                ora     b       ;
                mov     b,a     ;
                jmp     akloop  ;
;
notdigit:                       ;;
                mov     a,c     ;
                cpi     0-16    ;
                mov     a,b     ;
                rrc             ;
                jc      signrcl ;
                jnz     trysign ;
                inx     d       ;
                jmp     akloop  ;
trysign:                        ;
                mov     a,b     ;
                rlc             ;
                rlc             ;
                cc      converr ;
                mov     a,c     ;
                cpi     0-3     ;
                jnz     tryplus ;
                mvi     a,0c0h  ;
                ora     b       ;
                mov     b,a     ;
                inx     d       ;
                jmp     akloop  ;
tryplus:                        ;
                cpi     0-5     ;
                cnz     converr ;
                mvi     a,40h   ;
                ora     b       ;
                mov     b,a     ;
                inx     d       ;
                jmp     akloop  ;
;
;               BINDEC - CONVERT BINARY NUMBER TO DECIMAL ASCII STRING
;
bindec:                         ;
                push    b       ;
                push    h       ;
                lxi     b,0     ;
                push    h       ;
                dad     h       ;
                pop     h       ;
                jnc     div10k  ;
                mvi     a,45    ;
                stax    d       ;
```

```
        inr     b       ;
        inx     d       ;
        call    comp2   ;
div10k:                 ;
        xchg            ;
        shld    bufadr  ;
        xchg            ;
        lxi     d,10000 ;
        call    cnvt1dig;
        lxi     d,1000  ;
        call    cnvt1dig;
        lxi     d,100   ;
        call    cnvt1dig;
        lxi     d,10    ;
        call    cnvt1dig;
        mov     a,l     ;
        adi     48      ;
        inr     c       ;
        lhld    bufadr  ;
        xchg            ;
        stax    d       ;
        mov     a,c     ;
        add     b       ;
        inx     d       ;
        pop     h       ;
        pop     b       ;
        ret             ;
;
cnvt1dig:               ;
        call    edivmod ;
        xchg            ;
        mov     a,e     ;
        ora     c       ;
        rz              ;
        mov     a,e     ;
        adi     48      ;
        inr     c       ;
        xchg            ;
        lhld    bufadr  ;
        mov     m,a     ;
        inx     h       ;
        shld    bufadr  ;
        xchg            ;
        ret             ;
;
; contains programmable delay - HL=delay, SHLD TIME
; contains programmable sound - LENGTH: = duration
;                               COUNT:  = delay
;                               DRATE:  = decrement rate
;                               IRATE:  = increment rate
; DRIVER storage locations
;
retvec: ds      2
grchar: db      0
gbit:   db      0
rbit:   db      0
crtbit: db      0
graphx: db      0
kpad:   db      0
curco:  ds      2
```

```
linco:  ds      2       ;
linco1: ds      2       ;
boxco:  ds      2       ;
boxco1: ds      2       ;
bufadr: ds      2       ;
time:   db      01,01   ; HL=001.001
scrpnt: ds      2
oldstack:
        ds      64h
stack:
astor:  db      0       ; psw storage
length: ds      1       ; duration of sound.
count:  ds      2       ; delay rate.
repeat: ds      2       ; decrement rate.
memmap: ds      2000    ; leave room for 25 X 80 screen
;
;       PROGRAM STARTS HERE
;       TO ADD DRIVER PROGRAM:
;       A>DDT DRIVER19.HEX
;       -IXXXXXXXX.HEX     <-- YOUR PROGRAM
;       -R                 <-- DRIVER PROGRAM IS NOW
;                              ADDED TO YOUR PROGRAM
;                              JUMP VECTORS ARE AT
;                              DEFINED LOCATIONS AT
;                              THE BEGINNING OF THIS
;                              LISTING.
;       -^C
;       A>LOAD XX FNAME.COM    PROGRAM IS NOW READY TO RUN
;
begin:  equ     $       ; jump vector from DRIVER19 program
```