```
;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
```

```
 61: lf      equ     0ah
 62: bs      equ     8
 63: ap      equ     27h
 64: bel     equ     7
 65: callop  equ     0cdh    ; call op-code.
 66: jmpop   equ     0c3h    ; jmp op-code.
 67: ;
 68:         org     orgin
 69: start:  lxi     h,0
 70:         dad     sp
 71:         shld    oldstack
 72:         lxi     sp,stack
 73:         jmp     begin
 74: ;
 75: ;       JUMP VECTORS
 76: ;
 77:         jmp     cls     ; clear screen
 78:         jmp     zsmem   ; zero memory map
 79:         jmp     ceol    ; clear end of line
 80:         jmp     crlf    ; outputt carriage and linefeed
 81:         jmp     insertl ; insert line
 82:         jmp     deletel ; delete line
 83:         jmp     insertc ; enter insert char mode
 84:         jmp     deletec ; delete character
 85:         jmp     xinsertc; exit insert char mode
 86:         jmp     creport ; cursor position report
 87:         jmp     cup     ; cursor up
 88:         jmp     decrl   ; decrement cursor Y
 89:         jmp     cleft   ; cursor left
 90:         jmp     decrh   ; decrement cursor X
 91:         jmp     cright  ; cursor right
 92:         jmp     incrh   ; increment cursor X
 93:         jmp     cdown   ; cursor down
 94:         jmp     incrl   ; increment cursor Y
 95:         jmp     home    ; cursor home
 96:         jmp     process ; process escape sequence
 97:         jmp     hset    ; Heath set mode
 98:         jmp     clr25   ; clear 25th line
 99:         jmp     hrset   ; Heath reset mode
100:         jmp     output  ; output a char
101:         jmp     input   ; check for input
102:         jmp     wait    ; wait for character
103:         jmp     waitupr ; wait for character and convert to uppercase
104:         jmp     bentry  ; BDOS entry point
105:         jmp     savall  ; save all registers
106:         jmp     retall  ; retrieve all registers
107:         jmp     adj     ; adjust cursor position
108:         jmp     update  ; update memmap
109:         jmp     rsmem   ; read memmap
110:         jmp     ssmem   ; store char in memmap
111:         jmp     graphix ; enter graphics mode
112:         jmp     xgraphix; exit graphics mode
113:         jmp     reverse ; enter reverse video mode
114:         jmp     xreverse; exit reverse video mode
115:         jmp     cursoff ; cursor off
116:         jmp     curson  ; cursor on
117:         jmp     cursl   ; line cursor
118:         jmp     cursb   ; block cursor
119:         jmp     savecurs; save cursor position
120:         jmp     retcurs ; put cursor at saved position
```

```
121:            jmp     boot    ; reboot CP/M
122:            jmp     restore ; same as above
123:            jmp     grout   ; output single graphic character
124:            jmp     spmsg   ; print message from stack
125:            jmp     show    ; print message from HL
126:            jmp     delay   ; time delay
127:            jmp     hldiv   ; H/2, L/2
128:            jmp     vector  ; draw line
129:            jmp     box     ; draw box
130:            jmp     seeline ; checks status of line
131:            jmp     sound   ; creates sound through H8 speaker
132: ;
133: ; CLEAR DISPLAY
134: ;
135: cls:       push    h
136:            lxi     h,101h
137:            shld    curco
138:            call    adj     ; incase on 25th line
139:            mvi     a,hcd
140:            call    process
141:            lxi     h,memmap
142:            shld    scrpnt
143:            pop     h
144: zsmem:     push    h
145:            lxi     h,memmap
146:            lxi     d,2001
147: zsmem1:    mvi     m,0
148:            inx     h
149:            dcx     d
150:            mov     a,e
151:            ora     d
152:            jnz     zsmem1
153:            pop     h
154:            ret
155: ;
156: ceol:      push    h
157:            mvi     a,heol
158:            call    process
159:            lhld    curco
160:            xchg
161:            lhld    scrpnt
162: eolloop:   mvi     m,0
163:            inx     h
164:            dcr     d
165:            jnz     eolloop
166:            pop     h
167:            ret
168: crlf:      mvi     a,cr
169:            call    output
170:            mvi     a,lf
171:            call    output
172:            push    h
173:            lhld    curco
174:            mvi     h,1
175:            inr     l
176:            shld    curco
177:            pop     h
178:            ret
179: ;
180: insertl:
```

```
181:            mvi     a,hil
182:            jmp     process
183: ;
184: deletel:
185:            mvi     a,hdl
186:            jmp     process
187: ;
188: insertc:
189:            mvi     a,heim
190:            jmp     process
191: ;
192: deletec:
193:            mvi     a,hdch
194:            jmp     process
195: ;
196: xinsertc:
197:            mvi     a,herm
198:            jmp     process
199: ;
200: creport:                    ; cursor position report
201:            call    savall  ; save all registers
202:            mvi     a,hcpr  ;
203:            call    process ;
204: creport1:
205:            call    wait    ; get ESC
206:            cpi     esc     ; check it
207:            jnz     creport1; loop until ESC
208: creport2:
209:            call    wait    ; get "Y"
210:            cpi     'Y'     ; check it
211:            jnz     creport2;
212:            call    wait    ; get y
213:            sui     31      ; subtract offset
214:            sta     curco+1 ; cursor y storage
215:            call    wait    ; get x
216:            sui     31      ; subtract offset
217:            sta     curco   ; cursor x storage
218:            call    retall  ; retrieve all registers
219:            ret
220: ;
221: cup:      mvi     a,hcuu
222:            call    process
223: decrl:    call    savall
224:            lhld    curco
225:            dcr     l
226:            shld    curco
227:            lhld    scrpnt
228:            lxi     d,-80
229:            dad     d
230:            shld    scrpnt
231:            call    retall
232:            ret
233: ;
234: cleft:    mvi     a,hcub
235:            call    process
236: decrh:    call    savall
237:            lhld    curco
238:            dcr     h
239:            shld    curco
240:            lhld    scrpnt
```

```
241:            dcx     h
242:            shld    scrpnt
243:            call    retall
244:            ret
245: ;
246: cright:    mvi     a,hcuf
247:            call    process
248: incrh:     call    savall
249:            lhld    curco
250:            inr     h
251:            shld    curco
252:            lhld    scrpnt
253:            inx     h
254:            shld    scrpnt
255:            call    retall
256:            ret
257: ;
258: cdown:     mvi     a,hcud
259:            call    process
260: incrl:     call    savall
261:            lhld    curco
262:            inr     l
263:            shld    curco
264:            lhld    scrpnt
265:            lxi     d,80
266:            dad     d
267:            shld    scrpnt
268:            call    retall
269:            ret
270: ;
271: home:      push    h
272:            mvi     a,hcuh
273:            call    process
274:            lxi     h,101h
275:            shld    curco
276:            lxi     h,memmap
277:            shld    scrpnt
278:            pop     h
279:            ret
280: ;
281: process:
282:            push    psw
283:            mvi     a,esc
284:            call    output
285:            pop     psw
286:            jmp     output
287: hset:      push    psw
288:            mvi     a,hsm
289: hsetm:     call    process
290:            pop     psw
291:            call    output
292:            ret
293: ;
294: clr25:     mvi     a,'1'
295:            call    hrset
296:            mvi     a,'1'
297:            jmp     hset
298: ;
299: hrset:     push    psw
300:            mvi     a,hrm
```

```
301:            jmp     hsetm
302: ;
303: output:    mov     e,a
304:            mvi     c,direct
305:            jmp     bentry
306: output1:   lhld    retvec
307:            pchl
308: ;
309: input:     mvi     e,dinput
310:            mvi     c,direct
311:            jmp     bentry
312: ;
313: wait:      call    input
314:            ora     a
315:            jz      wait
316:            ret
317: ;
318: waitupr:
319:            call    wait
320:            cpi     'a'
321:            rc
322:            ani     5fh     ; make uppercase
323:            ret
324: ;
325: bentry:    push    b
326:            push    d
327:            push    h
328:            call    bdos
329:            pop     h
330:            pop     d
331:            pop     b
332:            ret
333: ;
334: savall:    xthl
335:            push    d
336:            push    b
337:            push    psw
338:            push    h
339:            ret
340: ;
341: retall:    pop     h
342:            pop     psw
343:            pop     b
344:            pop     d
345:            xthl
346:            ret
347: ;
348: adj:       push    h
349:            mvi     a,hdca
350:            call    process
351:            mov     a,l
352:            adi     offset
353:            call    output
354:            mov     a,h
355:            adi     offset
356:            call    output
357:            pop     h
358: adjmem:    push    h
359:            mov     b,h
360:            mov     c,l
```

```
361:            lxi     h,memmap
362:            lxi     d,50h
363:            dcr     c
364:            jz      adj1
365: adjloop:dad     d
366:            dcr     c
367:            jnz     adjloop
368: adj1:      mov     e,b
369:            mvi     d,0
370:            dad     d
371:            shld    scrpnt
372:            pop     h
373:            ret
374: ;
375: update:  push    h
376:            lhld    scrpnt
377:            mov     m,a
378:            inx     h
379:            shld    scrpnt
380:            pop     h
381:            ret
382: ;
383: rsmem:    push    h
384:            call    adjmem  ; 01/18/83
385:            lhld    scrpnt
386:            mov     a,m
387:            pop     h
388:            ret
389: ;
390: ssmem:    push    h
391:            push    psw     ; save character
392:            call    adjmem  ; 01/18/83
393:            lhld    scrpnt
394:            pop     psw     ; get character back
395:            mov     m,a
396:            pop     h
397:            ret
398: ;
399: graphix:
400:            mvi     a,hegm
401:            call    process
402:            mvi     a,1
403:            sta     gbit
404:            ret
405: ;
406: xgraphix:
407:            mvi     a,hxgm
408:            call    process
409:            mvi     a,0
410:            sta     gbit
411:            ret
412: ;
413: reverse:
414:            mvi     a,herv
415:            call    process
416:            mvi     a,1
417:            sta     rbit
418:            ret
419: ;
420: xreverse:
```

```
421:            mvi     a,hxrv
422:            call    process
423:            mvi     a,0
424:            sta     rbit
425:            ret
426: ;
427: cursoff:
428:            mvi     a,'x'
429:            call    process
430:            mvi     a,'5'
431:            jmp     output
432: ;
433: curson:
434:            mvi     a,'y'
435:            call    process
436:            mvi     a,'5'
437:            jmp     output
438: ;
439: cursl:    mvi     a,hrm
440:            call    process
441:            mvi     a,'4'
442:            jmp     output
443: ;
444: cursb:    mvi     a,hsm
445:            call    process
446:            mvi     a,'4'
447:            jmp     output
448: ;
449: savecurs:
450:            mvi     a,hscp
451:            jmp     process
452: ;
453: retcurs:
454:            mvi     a,hrcp
455:            jmp     process
456: ;
457: boot:
458: restore:
459:            mvi     a,hram
460:            call    process
461:            lhld    oldstack
462:            sphl                    ; set stack
463:            ret                     ; return to CCP
464: ;
465: grout:     sta     grchar  ; save character
466:            call    graphix ; graphics on.
467:            lda     grchar  ; get character
468:            call    output  ; output it.
469:            call    xgraphix;
470:            ret
471: ;
472: spmsg:     xthl                    ; SP=HL  HL=SP
473:            call    show    ; print message
474:            xthl                    ; SP=HL  HL=SP
475:            ret                     ; finished
476: ;
477: show:     mov     a,m
478:            cpi     '@'
479:            jz      printat
480:            cpi     '['
```

```
481:            jz      special
482:            cpi     0
483:            rz
484:            inx     h
485:            call    update
486:            call    output
487:            jmp     show
488: printat:
489:            inx     h
490:            mov     c,m
491:            inx     h
492:            mov     b,m
493:            inx     h
494:            call    savall
495:            mov     l,c
496:            mov     h,b
497:            call    adj
498:            call    retall
499:            jmp     show
500: special:
501:            inx     h
502:            mov     a,m
503:            cpi     'R'
504:            cz      reverse
505:            cpi     'r'
506:            cz      xreverse
507:            cpi     'G'
508:            cz      graphix
509:            cpi     'g'
510:            cz      xgraphix
511:            cpi     'S'
512:            cz      setmodes
513:            cpi     's'
514:            cz      rsetmodes
515:            cpi     'C'
516:            cz      cls
517:            cpi     'c'
518:            cz      ceol
519:            cpi     ']'
520:            jnz     special
521:            inx     h
522:            jmp     show
523: setmodes:
524:            mvi     a,hsm
525: setentry:
526:            call    process
527:            inx     h
528:            mov     a,m
529:            jmp     output
530: rsetmodes:
531:            mvi     a,hrm
532:            jmp     setentry
533: delay:     call    savall
534:            lhld    time
535: delay1:    dcx     h
536:            mov     a,h
537:            ora     l
538:            jnz     delay1
539:            call    retall
540:            ret
```

```
541: hldiv:  push    psw
542:         mov     a,h
543:         rar             ; H/2
544:         mov     h,a     ; H=H/2
545:         sbb     a       ; subtract with borrow
546:         mov     b,a     ; B=0ffh if carry, 0 if not
547:         mov     a,l
548:         rar             ; L/2
549:         mov     l,a     ; L=L/2
550:         sbb     a       ; get remainder
551:         mov     c,a     ; C=0ffh if carry
552:         pop     psw     ;
553:         ret
554: vector: lhld    linco   ; point #1
555:         xchg            ; to DE.
556:         lhld    linco1  ; point #2 to HL.
557:         lxi     b,0000h ; push stack empty flag
558:         push    b       ;
559:         mov     a,d     ;
560:         cmp     h       ;
561:         jc      noxg    ;
562:         jnz     xg      ;
563:         mov     a,e     ;
564:         cmp     l       ;
565:         jc      noxg    ;
566: xg:     xchg            ;
567: noxg:   mov     b,h     ;
568:         mov     c,l     ;
569:         lxi     h,inral ;
570:         mvi     m,3ch   ;
571:         mov     a,e     ;
572:         cmp     c       ;
573:         jc      loop    ;
574:         mvi     m,0ch   ;
575: ; origin = DE
576: ; endpt  = BC
577: ; midpt  = HL
578: loop:   mov     a,d     ;
579:         cmp     b       ;
580:         jz      eqx     ;
581:         push    b       ;
582:         add     b       ;
583:         rar             ;
584:         mov     b,a     ;
585:         inr     a       ;
586:         mov     h,a     ;
587:         mov     a,e     ;
588:         cmp     c       ;
589:         jz      eqy     ;
590: neqy:   add     c       ;
591:         rar             ;
592:         mov     c,a     ;
593: inral:  inr     a       ;
594: eqy:    mov     l,a     ;
595:         push    h       ;
596:         jmp     loop    ;
597: eqx:    mov     a,e     ;
598:         cmp     c       ;
599:         jz      eqxy    ;
600:         push    b       ;
```

```
601:            mov     h,d     ;
602:            jmp     neqy    ;
603: vplot:                     ; either (rsmem) or (plot)
604: eqxy:      call    plot    ;
605: eqxy1:     pop     d       ;
606:            mov     a,d     ;
607:            ora     e       ;
608:            jz      vexit   ; exit line routine
609:            pop     b       ;
610:            jmp     loop    ;
611: vexit:     mvi     a,0     ; finished flag
612:            ret             ; return
613: ;
614: ; PLOT routine plots points according
615: ; to the VECTOR routine.
616: ; PLOT pixels  48 X 160.  Where X = 1 to 160 & Y = 1 to 48
617: ;
618: plot:      xra     a       ; clear carry flag
619:            mov     a,e     ; get y
620:            rar             ; divide by 2
621:            mov     l,a     ; store in L
622:            jnc     plotop  ; plot top section
623:            xra     a       ; clear carry
624:            mov     a,d     ; get x
625:            rar             ; divide by 2
626:            mov     h,a     ; store in H
627:            jnc     plotbl  ; bottom left
628:            mvi     b,8     ; bottom right
629:            jmp     plot1   ; plot point
630: plotbl:    mvi     b,4     ; bottom left point
631:            jmp     plot1   ; plot point
632: plotop:    xra     a       ; carry=0
633:            mov     a,d     ; get x
634:            rar             ; divide by 2
635:            mov     h,a     ; store in H
636:            jnc     plot1   ; top left
637:            mvi     b,2     ; top right
638:            jmp     plot1   ; set point
639: plot1:     mvi     b,1     ; top left
640: plot1:     push    b       ; save BC
641:            call    rsmem   ; read memmap
642:            pop     b       ; get BC back
643:            sta     astor   ; save character
644:            ana     b       ; and with B
645:            rnz             ; bit already set
646:            lda     astor   ;
647:            ora     b       ; OR with B
648:            call    ssmem   ; update in memmap
649:            push    psw     ;
650:            call    adj     ;
651:            pop     psw     ;
652: plot2:     cpi     1       ; is it bit 1?
653:            jz      point1  ; set bit 1
654:            cpi     2       ; bit 2?
655:            jz      point2  ; set bit 2
656:            cpi     3       ; bit 1 & 2?
657:            jz      point3  ; set bit 1 & 2
658:            cpi     4       ; bit 4?
659:            jz      point4  ; set bit 4
660:            cpi     5       ; bit 4 & 1?
```

```
661:           jz      point5  ; set bit 4 & 1
662:           cpi     6       ; bit 4 & 2?
663:           jz      point6  ; set bit 4 & 2
664:           cpi     7       ; bit 1,2,4?
665:           jz      point7  ; set bit 1,2,4
666:           cpi     8       ; bit 8?
667:           jz      point8  ; set bit 8
668:           cpi     9       ; bit 8 & 1?
669:           jz      point9  ; set bit 8,1
670:           cpi     0ah     ; bit 8 & 2?
671:           jz      pointa  ; set bit 8,2
672:           cpi     0bh     ; bit 1,2,8?
673:           jz      pointb  ; set bit 1,2,8
674:           cpi     0ch     ; bit 8 & 4?
675:           jz      pointc  ; set bit 8,4
676:           cpi     0dh     ; bit 8,4,1?
677:           jz      pointd  ; set bit 1,4,8
678:           cpi     0eh     ; bit 8,4,2?
679:           jz      pointe  ; set bit 2,4,8
680: pointf:   call    reverse ; enter reverse video
681:           mvi     a,' '   ; bits 1,2,4,8
682:           call    output  ; output space
683:           jmp     xreverse; exit reverse video
684: pointe:   call    reverse ; enter reverse video
685:           mvi     a,'n'   ; bits 2,4,8
686:           call    grout   ; output graphics
687:           jmp     xreverse; exit reverse video
688: pointd:   call    reverse ; reverse video mode
689:           mvi     a,'o'   ; bits 1,4,8
690:           call    grout   ; output graphic char
691:           jmp     xreverse; exit reverse video
692: pointc:   call    reverse ; reverse video
693:           mvi     a,'p'   ;
694:           call    grout   ;
695:           jmp     xreverse;
696: pointb:   call    reverse ; reverse video
697:           mvi     a,'m'   ;
698:           call    grout   ;
699:           jmp     xreverse;
700: pointa:   mvi     a,'q'   ; bits 2,8
701:           jmp     grout   ; output graphic char
702: point9:   jmp     pointd  ; bits 1,8 +4
703: point8:   mvi     a,'l'   ; bit 8
704:           jmp     grout   ; output graphic char
705: point7:   call    reverse ; reverse video
706:           mvi     a,'l'   ; bits 1,2,4
707:           call    grout   ; graphic char
708:           jmp     xreverse; exit
709: point6:   jmp     pointe  ; bits 2,4
710: point5:   call    reverse ; reverse video
711:           mvi     a,'q'   ;
712:           call    grout   ;
713:           jmp     xreverse; exit
714: point4:   mvi     a,'m'   ; bit 4
715:           jmp     grout   ; output graphics
716: point3:   mvi     a,'p'   ; bits 1,2
717:           jmp     grout   ;
718: point2:   mvi     a,'o'   ; bit 2
719:           jmp     grout   ;
720: point1:   mvi     a,'n'   ; bit 1
```

```
721:            jmp     grout   ;
722: box:       lhld    boxco   ;
723:            shld    linco   ; x1,y1
724:            xchg            ;
725:            lhld    boxco1  ;
726:            mov     l,e     ; make y2=y1
727:            shld    linco1  ; x1,y1 -> x2,y1
728:            call    boxplot ; save all registers
729:            lhld    boxco1  ; get original x2,y2
730:            shld    linco1  ;
731:            mov     d,h     ; make x1=x2
732:            xchg            ;
733:            shld    linco   ; x2,y1 -> x2,y2
734:            call    boxplot ;
735:            lhld    boxco   ; get original x1,y1
736:            xchg            ;
737:            lhld    boxco1  ;
738:            mov     e,l     ; make y1=y2
739:            xchg            ;
740:            shld    linco   ; x2,y2 -> x1,y2
741:            call    boxplot ;
742:            lhld    boxco   ; get original x1,y1
743:            shld    linco   ;
744:            xchg            ;
745:            lhld    boxco1  ; original x2,y2
746:            mov     h,d     ; x1,y2 -> x1,y1
747:            shld    linco1  ;
748: boxplot:call    savall  ; save all registers
749:            call    vector  ; draw line
750:            call    retall  ; retrieve all regs
751:            ret             ; return for more
752: ;
753: ;       SEELINE - draws imaginary line and
754: ;                 checks line status.
755: ;
756: seeline:mvi     a,jmpop ; jump opcode.
757:            sta     eqxy    ; store in vector.
758:            lxi     h,rsmem ; look at memmap.
759:            shld    eqxy+1  ; store in vector.
760: seelin1:call    vector  ; check line.
761:            cpi     0       ; finished checking?
762:            jz      seelin2 ; exit
763:            cpi     ' '     ; empty space?
764:            jz      eqxy1   ; continue checking.
765:            jmp     eqxy1   ;
766: seelin2:mvi     a,callop; call opcode.
767:            sta     eqxy    ; put in vector.
768:            lxi     h,plot  ; plot line.
769:            shld    eqxy+1  ; store in vector.
770:            ret             ; finished checking.
771: ; SOUND -- creates sounds according to data in SNDMEM.
772: ;           Enter : DE = delay rate
773: ;
774: sound:  call    savall  ; save everything
775: sound0: mvi     a,10h   ; speaker on bit.
776:            out     0f0h    ; output to port 360Q
777: sound1: lda     length  ;
778:            dcr     a
779:            sta     length
780:            jnz     sound1  ; loop for good sound.
```

```
781:            mvi     a,0f4h  ; speaker off bit.
782:            out     0f0h    ; port 360Q
783:            lhld    count   ; delay.
784:            xchg            ; DE = delay from COUNT.
785: sound2:    dcx     d       ; decrement delay.
786:            mov     a,d     ;
787:            ora     e       ; test for zero.
788:            jnz     sound2  ;
789:            lhld    repeat  ; times to repeat
790:            mov     a,h
791:            ora     l
792:            jz      sound3  ; finished.
793:            dcx     h
794:            shld    repeat
795:            call    input   ; test for input
796:            cpi     03h     ; CTRL-C
797:            jz      sound3  ; exit
798:            jmp     sound0  ; repeat until HL=0000
799: sound3:    jmp     retall  ; return
800: ;
801: ;       16-bit math package from BYTE, May 1981 - vol 6 #5
802: ;
803: ;       EADD    (HL)=(HL)+(DE)
804: ;       ESUB    (HL)=(HL)-(DE)
805: ;       EMULT   (HL)=(HL)*(DE)
806: ;       EDIVMOD (HL)=(HL)/(DE), (DE)=(HL) MOD (DE)
807: ;       ESIGN   SET (S), (Z) FLAG TO REFLECT (HL)-(DE), LEAVING
808: ;               (HL) UNCHANGED.
809: ;       ECMP    SET (S), (Z) FLAGS TO REFLECT (HL), LEAVING (HL)
810: ;               AND (DE) UNCHANGED.
811: ;
812: ;       DECBIN  CONVERT ASCII CHARACTER STRING REPRESENTING A SIGNED
813: ;               DECIMAL INTEGER TO A SIGNED BINARY NUMBER.
814: ;       BINDEC  CONVERT A SIGNED BINARY NUMBER TO AN ASCII STRING
815: ;               REPRESENTING THE SIGNED DECIMAL VALUE OF THE NUMBER.
816: ;
817: ;       MATH PACKAGE EXECUTION TIMES IN MICRO-SECONDS
818: ;
819: ;       ROUTINE         TYPICAL         WORST CASE
820: ;
821: ;       EADD            30              54
822: ;       ESUB            50              74
823: ;       EMULT           370             517
824: ;       EDIVMOD         680             2500
825: ;
826: overflow        ret     ; Return if error
827: converr        ret     ; Same here
828: ;
829: eadd:                   ; 16-bit addition
830:            mov     a,h     ; test signs
831:            xra     d       ;
832:            ani     80h     ;
833:            dad     d       ; add, without affecting zero flag...
834:            jnz     esign   ; skip overflow test if signs differ
835:            rar             ; test for overflow by...
836:            xra     h       ; ...exclusive or of CY and sign of result
837:            ral             ;
838:            cc      overflow; check for arith overflow
839: ;
840: ;       ESIGN
```

```
841: ;
842: esign:                      ; set (S), (Z) flags to reflect (HL)
843:          xra     a         ; clear flags
844:          add     h         ; set flags to reflect HI byte
845:          rnz               ; return if HI byte is non-0
846:          add     l         ; else, see if (L) is 0 too...
847:          rz                ; and if so, return
848:          xra     a         ; else, force flags to show '+'
849:          inr     a         ;
850:          ret               ;
851: ;
852: ;        ESUB
853: ;
854: esub:                       ; 16-bit subtraction
855:          push    d         ;
856:          xchg              ;
857:          call    comp2     ; form 25 complement of subtrahend...
858:          call    eadd      ; ... and proceed as in addition
859:          pop     d         ;
860:          ret               ;
861: ;
862: ;        ECHS - CHANGE SIGN OF REGISTER (HL)
863: ;
864: echs:                       ;
865:          mov     a,h       ;
866:          sui     80h       ; check for that one nasty case...
867:          jnz     echsgo    ; ... of (HL) = -32768...
868:          add     l         ; ... which can't be complemented right
869:          cz      overflow  ; ... and when detected, abort
870: echsgo:                     ;
871:          call    comp2     ; else, form 25 complement in (HL)
872:          jmp     esign     ; set flags and return
873: ;
874: ;        25 COMPLEMENT OF (HL)
875: ;
876: comp2:                      ;
877:          mov     a,h       ;
878:          cma               ;
879:          mov     h,a       ;
880:          mov     a,l       ;
881:          cma               ;
882:          mov     l,a       ;
883:          inx     h         ;
884:          ret
885: ;
886: ;        EMULT
887: ;
888: emult:                      ;
889:          push    b         ;
890:          push    d         ;
891:          call    rsltsign  ; find result sign, abs val of operands
892:          xra     a         ;
893:          add     h         ;
894:          jz      hlsmall   ; branch if (HL) less than 8 bits
895:          xra     a         ;
896:          add     d         ; else, other op must be .lt. 8 bits...
897:          cnz     overflow  ; ... or overflow would result
898:          xchg              ;
899: hlsmall:                    ;
900:          mov     a,l       ; move 8-bit or less multiplier to (A)
```

```
901:            lxi     h,0       ; initialize partial product
902: xmloop:                      ;
903:            stc               ; clear carry
904:            cmc               ;
905:            rar               ; rotate multiplier right off end
906:            jnc     shiftop   ; if bit shifted-out was 0, skip
907:            dad     d         ; else, add multiplicand to partial prod.
908:            cc      overflow  ; ... while checking for overflow
909: shiftop:                     ;
910:            xchg              ;
911:            dad     h         ;
912:            cc      overflow  ;
913:            xchg              ;
914:            ora     a         ;
915:            jnz     xmloop    ;
916:            pop     d         ;
917: signrcl:                     ;
918:            mov     a,h       ;
919:            rlc               ;
920:            cc      overflow  ;
921:            mov     a,b       ;
922:            ral               ;
923:            cc      comp2     ;
924:            pop     b         ;
925:            jmp     esign     ;
926: ;
927: ;           COMPUTE SIGN OF RESULT FOR * AND /
928: ;
929: rsltsign:                    ;
930:            mov     b,h       ;
931:            mov     a,h       ;
932:            ral               ;
933:            cc      comp2     ;
934:            xchg              ;
935:            mov     a,h       ;
936:            xra     b         ;
937:            mov     b,a       ;
938:            mov     a,h       ;
939:            ral               ;
940:            jc      comp2     ;
941:            ret               ;
942: ;
943: ;           DIVIDE (HL) BY (DE)
944: ;
945: edivmod:                     ;
946:            push    b         ;
947:            xra     a         ;
948:            ora     e         ;
949:            ora     d         ;
950:            cz      overflow  ;
951:            call    rsltsign  ;
952:            mov     a,h       ;
953:            ora     d         ;
954:            rlc               ;
955:            cc      overflow  ;
956:            push    b         ;
957:            mov     c,e       ;
958:            mov     b,d       ;
959:            lxi     d,0       ;
960:            push    d         ;
```

```
 961:          xchg            ;
 962:          lxi     h,1     ;
 963: dbldiv:                  ;
 964:          dad     h       ;
 965:          xchg            ;
 966:          dad     h       ;
 967:          call    cmpbh   ;
 968:          xchg            ;
 969:          jnc     dbldiv  ;
 970:          xchg            ;
 971: halvediv:                 ;
 972:          xchg            ;
 973:          call    divby2  ;
 974:          jz      divdone ;
 975:          xchg            ;
 976:          call    divby2  ;
 977:          call    cmpbh   ;
 978:          jm      halvediv;
 979:          mov     a,c     ;
 980:          sub     l       ;
 981:          mov     c,a     ;
 982:          mov     a,b     ;
 983:          sbb     h       ;
 984:          mov     b,a     ;
 985:          xthl            ;
 986:          dad     d       ;
 987:          xthl            ;
 988:          jmp     halvediv;
 989: divdone:                  ;
 990:          pop     h       ;
 991:          mov     e,c     ;
 992:          mov     d,b     ;
 993:          pop     b       ;
 994:          jmp     signrcl ;
 995: cmpbh:                    ;
 996:          mov     a,c     ;
 997:          sub     l       ;
 998:          mov     a,b     ;
 999:          sbb     h       ;
1000:          ret             ;
1001: divby2:                  ;
1002:          xra     a       ;
1003:          mov     a,h     ;
1004:          rar             ;
1005:          mov     h,a     ;
1006:          mov     a,l     ;
1007:          rar             ;
1008:          mov     l,a     ;
1009:          ora     h       ;
1010:          ret             ;
1011: ;
1012: ;        DECBIN - CONVERT ASCII DECIMAL TO BINARY NUMBER
1013: ;
1014: decbin:                   ;
1015:          push    b       ;
1016:          mvi     b,0     ;
 017:          lxi     h,0     ;
1018: akloop:                  ;
1019:          ldax    d       ;
1020:          sui     48      ;
```

```
1021:           mov     c,a     ;
1022:           jm      notdigit;
1023:           cpi     10      ;
1024:           jp      notdigit;
1025:           push    d       ;
 026:           lxi     d,10    ;
1027:           call    emult   ;
1028:           mvi     d,0     ;
1029:           mov     e,c     ;
1030:           call    eadd    ;
1031:           pop     d       ;
1032:           inx     d       ;
1033:           mvi     a,1     ;
1034:           ora     b       ;
1035:           mov     b,a     ;
1036:           jmp     akloop  ;
1037: ;
1038: notdigit:                 ;;
1039:           mov     a,c     ;
1040:           cpi     0-16    ;
1041:           mov     a,b     ;
1042:           rrc             ;
1043:           jc      signrcl ;
1044:           jnz     trysign ;
1045:           inx     d       ;
1046:           jmp     akloop  ;
1047: trysign:                  ;
1048:           mov     a,b     ;
1049:           rlc             ;
1050:           rlc             ;
1051:           cc      converr ;
1052:           mov     a,c     ;
1053:           cpi     0-3     ;
1054:           jnz     tryplus ;
1055:           mvi     a,0c0h  ;
1056:           ora     b       ;
1057:           mov     b,a     ;
1058:           inx     d       ;
1059:           jmp     akloop  ;
1060: tryplus:                  ;
1061:           cpi     0-5     ;
1062:           cnz     converr ;
1063:           mvi     a,40h   ;
1064:           ora     b       ;
1065:           mov     b,a     ;
1066:           inx     d       ;
1067:           jmp     akloop  ;
1068: ;
1069: ;       BINDEC - CONVERT BINARY NUMBER TO DECIMAL ASCII STRING
1070: ;
1071: bindec:                   ;
1072:           push    b       ;
1073:           push    h       ;
1074:           lxi     b,0     ;
1075:           push    h       ;
1076:           dad     h       ;
 077:           pop     h       ;
1078:           jnc     div10k  ;
1079:           mvi     a,45    ;
1080:           stax    d       ;
```

```
1081:           inr     b       ;
1082:           inx     d       ;
1083:           call    comp2   ;
1084: div10k:                   ;
1085:           xchg            ;
 086:           shld    bufadr  ;
1087:           xchg            ;
1088:           lxi     d,10000 ;
1089:           call    cnvt1dig;
1090:           lxi     d,1000  ;
1091:           call    cnvt1dig;
1092:           lxi     d,100   ;
1093:           call    cnvt1dig;
1094:           lxi     d,10    ;
1095:           call    cnvt1dig;
1096:           mov     a,l     ;
1097:           adi     48      ;
1098:           inr     c       ;
1099:           lhld    bufadr  ;
1100:           xchg            ;
1101:           stax    d       ;
1102:           mov     a,c     ;
1103:           add     b       ;
1104:           inx     d       ;
1105:           pop     h       ;
1106:           pop     b       ;
1107:           ret             ;
1108: ;
1109: cnvt1dig:                 ;
1110:           call    edivmod ;
1111:           xchg            ;
1112:           mov     a,e     ;
1113:           ora     c       ;
1114:           rz              ;
1115:           mov     a,e     ;
1116:           adi     48      ;
1117:           inr     c       ;
1118:           xchg            ;
1119:           lhld    bufadr  ;
1120:           mov     m,a     ;
1121:           inx     h       ;
1122:           shld    bufadr  ;
1123:           xchg            ;
1124:           ret             ;
1125: ;
1126: ; contains programmable delay - HL=delay, SHLD TIME
1127: ; contains programmable sound - LENGTH: = duration
1128: ;                               COUNT:  = delay
1129: ;                               DRATE:  = decrement rate
1130: ;                               IRATE:  = increment rate
1131: ; DRIVER storage locations
1132: ;
1133: retvec: ds      2
1134: grchar: db      0
1135: gbit:   db      0
1136: rbit:   db      0
 137: crtbit: db      0
1138: graphx: db      0
1139: kpad:   db      0
1140: curco:  ds      2
```

```
1141: linco:   ds      2       ;
1142: linco1:  ds      2       ;
1143: boxco:   ds      2       ;
1144: boxco1:  ds      2       ;
1145: bufadr:  ds      2       ;
 146: time:    db      01,01   ; HL=001.001
1147: scrpnt:  ds      2
1148: oldstack:
1149:          ds      64h
1150: stack:
1151: astor:   db      0       ; psw storage
1152: length:  ds      1       ; duration of sound.
1153: count:   ds      2       ; delay rate.
1154: repeat:  ds      2       ; decrement rate.
1155: memmap:  ds      2000    ; leave room for 25 X 80 screen
1156: ;
1157: ;       PROGRAM STARTS HERE
1158: ;       TO ADD DRIVER PROGRAM:
1159: ;       A>DDT DRIVER19.HEX
1160: ;       -IXXXXXXXX.HEX      <-- YOUR PROGRAM
1161: ;       -R                  <-- DRIVER PROGRAM IS NOW
1162: ;                               ADDED TO YOUR PROGRAM
1163: ;                               JUMP VECTORS ARE AT
1164: ;                               DEFINED LOCATIONS AT
1165: ;                               THE BEGINNING OF THIS
1166: ;                               LISTING.
1167: ;       -^C
1168: ;       A>LOAD XX FNAME.COM    PROGRAM IS NOW READY TO RUN
1169: ;
1170: begin:  equ     $       ; jump vector from DRIVER19 program
```