```
                        1
                        2   ;       BCA/l/e Z-IOP.bca;pnew z-iop.mb@P@
                        3   ;       Current version is version 1.5
                        4
                        5   ;       Last change:
                        6
                        7   ;       moved interrupt vectors to RAM to allow disk
                        8   ;       programs to use interrupts not used by rom
                        9
                       10
                       11   ;       This file contains the code that will be resident
                       12   ;       in the IOP rom with BOOTABLE disks--Interim arrangement
                       13   ;       Author: Bruce Horn
                       14   ;       Last changed: June 18, 1979  3:19 PM
                       15
                       16
                       17           .Predefine      "8086Predefs.sr"
                       18
                       19           .Get    "Z-IOPRamDefs.bca"      ;Ram locations...
                        1   ;       Z-IOPRamDefs.bca
                        2
                        3   ;       This file contains the IOP Ram locations and symbols.
                        4   ;       The IOP can address ram beginning at 1000H
                        5   ;       All state is booted from the disk
                        6
                        7   ;       Author: Bruce Horn
                        8   ;       Last changed: June 18, 1979  3:04 PM
                        9
                       10
                       11   ;RAM definitions follow--
                       12
                       13   ;The following is PRIVATE to the IO processor
                       14
1000                   15   IOPRam=         1000            ;IOP Ram begins at this address
                       16
1000                   17   BitsGone=       IOPRam          ;Kbd Handler temp
1002                   18   Count=          BitsGone+2.     ;temp
1004                   19   Mask=           Count+2.        ;temp
1006                   20   Offset=         Mask+2.         ;temp (Meta keys modify this)
1008                   21   Temp1=          Offset+2.       ;Temp for interrupt routine
100A                   22   IRR=            Temp1+2.        ;Interrupt mask (IntsOn/Off)
100C                   23   Port100=        IRR+2.          ;port100 settings
100E                   24   Port60=         Port100+2.      ;Port60 settings
1010                   25   Port20=         Port60+2.       ;port20 settings
1012                   26   KbdOffset=      Port20+2.       ;current byte address on input
1014                   27   OldKbdData=     KbdOffset+2.    ;6 word keyboard input, old
                       28
1020                   29   R0=             OldKbdData+12.  ;various temporary registers
1022                   30   R1=             R0+2.           ;for rom to use
1024                   31   R2=             R1+2.
1026                   32   R3=             R2+2.
1028                   33   BitErrors=      R3+2.           ;following 8 bytes are bit errors
                       34                                   ;from mem test
                       35
1030                   36   .ShowNewCursor= BitErrors+8.    ;Other useful rom routines
1032                   37   .DisplayOn=     .ShowNewCursor+2.
1034                   38   .DisplayOff=    .DisplayOn+2.
1036                   39   .WaitBug=       .DisplayOff+2.
1038                   40   .WaitNoBug=     .WaitBug+2.
103A                   41   .IntsOn=        .WaitNoBug+2.
103C                   42   .IntsOff=       .IntsOn+2.
103E                   43   .SetP60=        .IntsOff+2.
1040                   44   .IOInit=        .SetP60+2.
1042                   45   DStatus=        .IOInit+2.      ;disk status on fail
1044                   46   JumpTo=         DStatus+2.      ;Byte/Seg address of init routine
1048                   47   NextDisk=       JumpTo+4.       ;another disk after this one?
104A                   48   ReTry=          NextDisk+2.     ;Retry counts/sector r/w
104C                   49   BeginAdr=       ReTry+2.        ;Begin loading address
104E                   50   BeginSeg=       BeginAdr+2.     ;Begin loading segment
1050                   51   OldAdr=         BeginSeg+2.     ;Last address since R/W sector
1052                   52   OldSeg=         OldAdr+2.       ;Last segment
1054                   53   Blocks=         OldSeg+2.       ;Blocks to read
1056                   54   Track=          Blocks+2.       ;Current track
1057                   55   Sector=         Track+1.        ;Current sector
1058                   56   .Buffer=        Sector+1.       ;location of disk buffer
105A                   57   Buffer=         .Buffer+2.      ;128 char disk buffer
```

```
                        58
                        59      ;The following is PUBLIC information--The emulator processor can
                        60      ;address this information, and possibly alter it
                        61
2000                    62      EPRam=          2000            ;EP can address ram starting here
                        63
                        64      ;Routines available in ROM
                        65
2000                    66      .BitBlt=        EPRam           ;address of BitBLT routine
2002                    67      .SetP100=       .BitBlt+2.      ;address of port100 routine
2004                    68      .SetP20=        .SetP100+2.     ;port 20
2006                    69      .InitDC=        .SetP20+2.      ;display controller init
2008                    70      .HideCursor=    .InitDC+2.      ;hides, locks cursor
200A                    71      .ShowCursor=    .HideCursor+2.  ;unlocks, shows cursor
200C                    72      .RomInit=       .ShowCursor+2.  ;initializes rom
                        73
                        74      ;Other reserved locations
                        75
200E                    76      QByte=          .RomInit+2.     ;TRB Queue Header
2010                    77      QSeg=           QByte+2.        ;segment of last block
2012                    78      QLock=          QSeg+2.         ;Queue locked-->negative
2014                    79      ICount=         QLock+2.        ;60hz interrupt counter
2016                    80      OldICount=      ICount+2.       ;old interrupt count for change
2018                    81      KbdStatus=      OldICount+2.    ;#chars in buffer|mouse buttons
201A                    82      MouseX=         KbdStatus+2.    ;Mouse X position
201C                    83      MouseY=         MouseX+2        ;Mouse Y position
201E                    84      RMX=            MouseY+2.       ;  Defines restricted rectangle
2020                    85      RMY=            RMX+2.          ;  for mouse--X, Y, Width, Height
2022                    86      RMW=            RMY+2.          ;  Mouse cannot move outside
2024                    87      RMH=            RMW+2.          ;  this rectangle
2026                    88      OldMX=          RMH+2.          ;old MouseX since cursor
2028                    89      OldMY=          OldMX+2.        ;Old MouseY since cursor
202A                    90      KbdData=        OldMY+2.        ;Keyboard data from Kbd UART (6w)
202B                    91      MYDelta=        KbdData+1.      ;Mouse Y delta
202C                    92      MXDelta=        KbdData+2.      ;Mouse X delta
202D                    93      KW0=            KbdData+3.      ;Keyboard word zero
202F                    94      KW1=            KbdData+5.      ;Keyboard word one
2031                    95      KW2=            KbdData+7.      ;Keyboard word two
2033                    96      KW3=            KbdData+9.      ;Keyboard word three
2036                    97      ReadPtr=        KbdData+12.     ;Read Pointer for kbd buffer
2038                    98      WritePtr=       ReadPtr+2.      ;Write Pointer
203A                    99      KbdBuffer=      WritePtr+2.     ;32 character kbd buffer
207A                   100      IntKey=         KbdBuffer+64.   ;has an interrupt key been hit?
207C                   101      DCInits=        IntKey+2.       ;Display controller init table
2094                   102      CDF=            DCInits+24.     ;Current Disk Format
20D0                   103      ADF=            CDF+60.         ;Alternate Disk Format
210C                   104      NIChars=        ADF+60.         ;number of interrupt keys
210E                   105      ICList=         NIChars+2.      ;[char],[char]...[char] <=8
211E                   106      .Cursor=        ICList+16.      ;2 byte oop
2120                   107      .Display=       .Cursor+2.      ;2 byte oop of display
2122                   108      .CursorTable=   .Display+2.     ;2 word addr. to cursor bb table
2126                   109      .BBTable=       .CursorTable+4. ;2 word addr. to display bb table
212A                   110      .KbdMap=        .BBTable+4.     ;2 word address to Keyboard Map
212E                   111      CTHeader=       .KbdMap+4.      ;TRB header for Cursor table
2136                   112      CursorTable=    CTHeader+8.     ;Cursor bitblt table
2136                   113      DisplayBits=    CursorTable+0.  ;display bitmap loc in table
213E                   114      CursorX=        CursorTable+8.  ;Cursor X position
2140                   115      CursorY=        CursorX+2       ;Cursor Y position
214E                   116      CursorBits=     CursorTable+24. ;cursor bitmap loc in table
215E                   117      CursorXOff=     CursorTable+40. ;cursor x offset
2160                   118      CursorYOff=     CursorXOff+2.   ;cursor y offset
2162                   119      CursorLock=     CursorYOff+2.   ;should cursor change?
2164                   120      BBHeader=       CursorTable+46. ;TRB header for BB table
216C                   121      BBTable=        BBHeader+8.     ;Display bitblt table
                       122
22A0                   123      RamInts=        22A0
                       124
22A0                   125      IR00=           RamInts         ;ParErrInt
22A5                   126      IR01=           IR00+5.         ;IPSysInt
22AA                   127      IR02=           IR01+5.         ;DiskInt
22AF                   128      IR03=           IR02+5.         ;EIAInt
22B4                   129      IR04=           IR03+5.         ;OddInt
22B9                   130      IR05=           IR04+5.         ;ADCInt
22BE                   131      IR06=           IR05+5.         ;KbdInt
22C3                   132      IR07=           IR06+5.         ;VSyncInt
                       133
```

```
                    134
                    135     ;The following are some useful constants that should be listed
                    136     ;as external:
                    137
                    138
0000                139     CurrentDS=      0               ;current code, data segment
0002                140     XorBB=          02              ;bitblt XOR
FFFF                141     Black=          0FFFF           ;Black
0000                142     White=          0               ;White for bitblt store constant
000C                143     StcBB=          0C              ;Store constant
                    144
                    145     ;BitBLT table offsets--
                    146
0000                147     BBDB=           0.              ;BitBLT Destination Byte address
0002                148     BBDS=           2.              ;BB Destination Segment address
0004                149     DRast=          4.              ;Destination raster
0006                150     DField=         6.              ;Destination field size
0008                151     DestX=          8.              ;Destination X
000A                152     DestY=          10.             ;Destination Y
000C                153     DestW=          12.             ;Destination Width
000E                154     DestH=          14.             ;Destination Height
0010                155     ClipX=          16.             ;Clip X
0012                156     ClipY=          18.             ;Clip Y
0014                157     ClipW=          20.             ;Clip Width
0016                158     ClipH=          22.             ;Clip Height
0018                159     BBSB=           24.             ;BitBLT Source Byte address
001A                160     BBSS=           26.             ;BitBlt Source Segment address
001C                161     SRast=          28.             ;Source raster
001E                162     SField=         30.             ;Source Field size
0020                163     SourceX=        32.             ;Source X
0022                164     SourceY=        34.             ;Source Y
0024                165     Function=       36.             ;Function
0026                166     GrayBits=       38.             ;Gray bits
                    167
                    168
                    169     ;Port100 Fields: (Disk/ADC selects)
                    170
                    171
FFFC                172     DiskPower=      0FFFC           ;+5=2, +12=1
FFFB                173     DiskSide=       0FFFB           ;Side zero/one
FFC7                174     DiskDrive=      0FFC7           ;D1=4, D2=2, D3=1
FE3F                175     ADCDevice=      0FE3F           ;Device select
F1FF                176     BitClk=         0F1FF           ;Bit clock field
EFFF                177     DiskMR=         0EFFF           ;Disk Controller master reset
DFFF                178     GoWordClock=    0DFFF           ;StopWordClock'
3FFF                179     ADCSpeed=       03FFF           ;ADC sampling rate
                    180
                    181     ;Port60 Fields: (DAC/Tablet selects)
                    182
FCFF                183     TabletSelect=   0FCFF           ;2=Tablet X, 1=Tablet Y
E3FF                184     DACSpeed=       0E3FF           ;DAC sampling rate
BFFF                185     SampleHoldA=    0BFFF           ;Sample/Hold channel A
DFFF                186     SampleHoldB=    0DFFF           ;Sample/Hold channel B
7FFF                187     SampleNotHold=  07FFF           ;Sample/Hold'
                    188
                    189
                    190     ;Port20 Fields: (IOP Processor selects)
                    191
FFF0                192     LEDSOn=         0FFF0           ;Light LEDs.  1111 lights all
FFEF                193     DisableRom=     0FFEF           ;Run out of main mem only
FFDF                194     CharCtr=        0FFDF           ;Char counter
FFBF                195     IOPLock=        0FFBF           ;IOP Processor lock
FF7F                196     BootSeqDone=    0FF7F           ;Stop mapping mem req into Rom
                    197
                    198     ;Ports and other port fields
0000                199     ic=             0               ;interrupt controller
0002                200     ocw1=           2               ;interrupt controller word1
00EF                201     ir4=            0EF             ;defines OddInt field
00BF                202     ir6=            0BF             ;defines KbdInt field
                    203
                    204
                    205
                    206             .END
                     20
                     21             .Get    "Z-Boot.bca"            ;debugger
                      1     ;       Z-Boot.BCA
```

```
                        2
                        3   ;       This file contains the boot code to enable the IOP to boot
                        4   ;       from the minifloppy.  It calls routines in Z-DRW.bca and
                        5   ;       Z-Subs.bca to read/write the disk
                        6   ;       Author: Bruce Horn
                        7   ;       Last edited: June 18, 1979  3:18 PM
                        8
                        9   ; segments
                       10
0000                   11   Ram = 0
0000                   12   Rom = 0
                       13
                       14   ; mem test range
                       15
1000                   16   first =         1000    ;low boundary
3000                   17   past =          3000    ;high boundary
2E00                   18   FirstSP=        2E00
                       19
                       20   ;8086 Interrupt vector
                       21
0000                   22           .loc 0          ;interrupt vector (if sys=0)
                       23
0000 9F 01             24   .adr DivErr             ;type 0 divide error
0002 00 00             25   .adr ram
0004 B7 01             26   .adr SST                ;type 1 single step
0006 00 00             27   .adr ram
0008 CF 01             28   .adr NMI                ;type 2 nmi
000A 00 00             29   .adr ram
000C E7 01             30   .adr BPT                ;type 3 breakpoint
000E 00 00             31   .adr ram
0010 FF 01             32   .adr OVF                ;type 4 overflow
0012 00 00             33   .adr ram
                       34
                       35   ;8259 Interrupt vector
                       36
0080                   37           .loc 4*020
                       38
0080 A0 22             39   .adr IR00               ;ParErrInt
0082 00 00             40   .adr Ram
0084 A5 22             41   .adr IR01               ;IPSysInt
0086 00 00             42   .adr Ram
0088 AA 22             43   .adr IR02               ;DiskInt
008A 00 00             44   .adr Ram
008C AF 22             45   .adr IR03               ;EIAInt
008E 00 00             46   .adr Ram
0090 B4 22             47   .adr IR04               ;OddInt
0092 00 00             48   .adr Ram
0094 B9 22             49   .adr IR05               ;ADCInt
0096 00 00             50   .adr Ram
0098 BE 22             51   .adr IR06               ;KbdInt
009A 00 00             52   .adr Ram
009C C3 22             53   .adr IR07               ;VSyncInt
009E 00 00             54   .adr Ram
                       55
                       56
                       57   IntVectors:
                       58
00A0 EA A7 07 00 00    59   jmps    ParErrInt Rom
00A5 EA BF 07 00 00    60   jmps    IPSysInt Rom
00AA EA D7 07 00 00    61   jmps    DiskInt Rom
00AF EA EF 07 00 00    62   jmps    EIAInt Rom
00B4 EA 46 07 00 00    63   jmps    OddInt Rom
00B9 EA 07 08 00 00    64   jmps    ADCInt Rom
00BE EA 76 07 00 00    65   jmps    KbdInt Rom
00C3 EA 1F 08 00 00    66   jmps    VSyncInt Rom
                       67
0FF0                   68           .loc 0FF0       ; reset vector
                       69
0FF0 EA D0 00 00 00    70   jmps restart rom        ; sets CS=F000(0)
                       71
00D0                   72           .loc 0D0
                       73
                       74   ;********Boot********
                       75
                       76   Restart:
                       77
```

```
                        78  ;First, disable interrupts and zero all segment registers
00D0 FA                 79          cli
00D1 B8 00 00           80          mov     ax,#0
00D4 8E D0              81          mov     ss,ax
00D6 8E D8              82          mov     ds,ax
00D8 8E C0              83          mov     es,ax
                        84  ;Stop mapping all mem requests into Rom--
00DA B0 88              85          mov     al,#88  ;BootSequenceDone       and
00DC BA 20 00           86          mov     dx,#20
00DF EE                 87          outd            ;ErrorCorrectionOff
00E0 BB 00 00           88          mov     bx,#0   ;start dispatch with 2
00E3 EB 08              89          j       waitSec
                        90
                        91  X:
00E5 00 00              92          .adr    0       ;not used
00E7 04 01              93          .adr    plus5
00E9 0D 01              94          .adr    plus12
00EB 16 01              95          .adr    start   ;ramtest label
                        96
                        97  waitsec:
00ED B9 01 00           98          mov     cx,#1
00F0 B8 00 70           99          mov     ax,#7000
                       100  LP:
00F3 FF C8             101          dec     ax
00F5 75 FC             102          jnz     LP
00F7 B8 00 70          103          mov     ax,#7000
00FA E2 F7             104          loop    LP
00FC FF C3             105          inc     bx
00FE FF C3             106          inc     bx
0100 FF A7 E5 00       107          jmpi    X!bx    ;jump to routine
                       108
                       109  plus5:
0104 B8 02 00          110          mov     ax,#2   ;+5 volts on
0107 BA 00 01          111          mov     dx,#100
010A EF                112          outdw
010B EB E0             113          j       waitSec
                       114  plus12:
010D B8 03 00          115          mov     ax,#3   ;+12 volts on
0110 BA 00 01          116          mov     dx,#100
0113 EF                117          outdw
0114 EB D7             118          j       waitSec
                       119
                       120  ;Test low ram for errors--
                       121
0116 B2 00             122  start:  mov     dl,#0           ;test number in dl
0118 BF 00 00          123          mov     di,#0           ;and offset in di
                       124
011B FE C2             125  again:  inc     dl              ;on to the next test
011D FF C7             126          inc     di              ;and the next offset
011F FF C7             127          inc     di              ;incremented by 2 for word
0121 80 FA 01          128          cmp     dl,#1           ;dispatch on dl
0124 74 1C             129           je     Xtest           ;it fits
0126 80 FA 02          130          cmp     dl,#2           ;test 2
0129 74 17             131           je     Xtest           ;it fits
012B 80 FA 03          132          cmp     dl,#3           ;test 3
012E 74 12             133           je     Xtest           ;it fits
0130 80 FA 04          134          cmp     dl,#4           ;test 4
0133 74 0D             135           je     Xtest           ;it fits
0135 E9 39 00          136          jmp     BootIt          ;assume the RAM works
                       137
                       138
                       139  BitsTable:
0138 00 00             140          .adr    0       ;not used
013A FF FF             141          .adr    0FFFF   ;Ones
013C 00 00             142          .adr    00000   ;Zeroes
013E AA AA             143          .adr    0AAAA   ;AAAA
0140 55 55             144          .adr    05555   ;5555
                       145
                       146  ;write bits in location and read back
0142 BE 00 10          147  Xtest:   mov     si,#first       ;ram address ← 0
0145 8B 9D 38 01       148          mov     bx,BitsTable!di
0149 89 1C             149  l3:     mov     0!si,bx         ;write in bits
014B FF C6             150          inc     si
014D FF C6             151          inc     si
014F 81 FE 00 30       152          cmp     si,#past        ;end or ram
0153 72 F4             153          jb      l3              ;llop if below
```

```
                                154
0155 B9 00 00                   155             mov     cx,#0           ;mask of which bits are bad
0158 BE 00 10                   156             mov     si,#first       ;ram address 0
015B 8B 04                      157     14:     mov     ax,0!si         ;read back ram location contents
015D 33 C3                      158             xor     ax,bx           ;any bits different?
015F 0B C8                      159             or      cx,ax           ;or them into mask
0161 FF C6                      160             inc     si
0163 FF C6                      161             inc     si
0165 81 FE 00 30                162             cmp     si,#past        ;end of ram
0169 72 F0                      163             jb      14              ;next
                                164
016B 89 8D 28 10                165             mov     BitErrors!di,cx ;Save error log in low mem
016F EB AA                      166             jmp     again           ; start all over
                                167
                                168
                                169     ;The ram works, so now boot from the disk
                                170
                                171     BootIt:
                                172
0171 BC 00 2E                   173             mov     sp,#firstSP     ;initialize SP
                                174
0174 B8 00 00                   175             mov     ax,#0           ;zero flags
0177 50                         176             push    ax
0178 9D                         177             popf
0179 FC                         178             cld
017A BE A0 00                   179             mov     si,#IntVectors
017D BF A0 22                   180             mov     di,#RamInts
0180 B9 28 00                   181             mov     cx,#40.
0183 F2                         182             rep
0184 A4                         183             movb                    ;BLT up interrupt tables
0185 E8 78 0A                   184             call    RomInit
0188 E8 7C 08                   185             call    LoadFloppy
018B 2E                         186             seg     cs
018C C7 06 46 10 00 00          187             mov     JumpTo+2,##0    ;set segment to zero
                                188     RunAgain:
0192 2E                         189             seg     cs
0193 FF 1E 44 10                190             callis  JumpTo          ;call the routine
0197 E8 A0 04                   191             call    WaitNoBug       ;and in case it returns..
019A E8 94 04                   192             call    WaitBug         ;wait for no bug first, then
019D EB F3                      193             jmp     RunAgain        ;run prog again on next bug
                                194
                                195     DivErr:
019F B8 00 00                   196             mov     ax,#0
01A2 BB F0 FF                   197             mov     bx,#LedsOn
01A5 E8 B8 04                   198             call    SetP20
01A8 E8 6D 04                   199             call    IntsOn
01AB E8 7B 04                   200             call    DisplayOn
01AE 2E                         201             seg     cs
01AF A1 A9 0D                   202             mov     ax,.ErrorCursor
01B2 E8 8E 04                   203             call    ShowNewCursor
01B5 EB FE                      204     H0:     j       H0
                                205
                                206     SST:
01B7 B8 01 00                   207             mov     ax,#1
01BA BB F0 FF                   208             mov     bx,#LedsOn
01BD E8 A0 04                   209             call    SetP20
01C0 E8 55 04                   210             call    IntsOn
01C3 E8 63 04                   211             call    DisplayOn
01C6 2E                         212             seg     cs
01C7 A1 A9 0D                   213             mov     ax,.ErrorCursor
01CA E8 76 04                   214             call    ShowNewCursor
01CD EB FE                      215     H1:     j       H1
                                216
                                217     NMI:
01CF B8 02 00                   218             mov     ax,#2
01D2 BB F0 FF                   219             mov     bx,#LedsOn
01D5 E8 88 04                   220             call    SetP20
01D8 E8 3D 04                   221             call    IntsOn
01DB E8 4B 04                   222             call    DisplayOn
01DE 2E                         223             seg     cs
01DF A1 A9 0D                   224             mov     ax,.ErrorCursor
01E2 E8 5E 04                   225             call    ShowNewCursor
01E5 EB FE                      226     H2:     j       H2
                                227
                                228     BPT:
01E7 B8 03 00                   229             mov     ax,#3
```

```
01EA BB F0 FF            230             mov     bx,#LedsOn
01ED E8 70 04            231             call    SetP20
01F0 E8 25 04            232             call    IntsOn
01F3 E8 33 04            233             call    DisplayOn
01F6 2E                  234             seg     cs
01F7 A1 A9 0D            235             mov     ax,.ErrorCursor
01FA E8 46 04            236             call    ShowNewCursor
01FD EB FE               237     H3:     j       H3
                         238
                         239     OVF:
01FF B8 04 00            240             mov     ax,#4
0202 BB F0 FF            241             mov     bx,#LedsOn
0205 E8 58 04            242             call    SetP20
0208 E8 0D 04            243             call    IntsOn
020B E8 1B 04            244             call    DisplayOn
020E 2E                  245             seg     cs
020F A1 A9 0D            246             mov     ax,.ErrorCursor
0212 E8 2E 04            247             call    ShowNewCursor
0215 EB FE               248     H4:     j       H4
                         249
                         250
                         251             .END
                          22
                          23             .Get    "Z-BitBLT.bca"          ;bitblt code
                           1     ;       Z-BitBLT.bca
                           2     ;       This file contains BitBLT for the 8086
                           3     ;       Version 2.0
                           4     ;       Author: Dan Ingalls
                           5     ;       Last changed: March 8, 1979  3:54 PM
                           6     ;        - fixed minx/y clipping
                           7     ;        - fixed destination segment
                           8     ;        - saves/restores DS
                           9     ;        - uses fast shift
                          10     ;
                          11     ;
0002                      12     DESBYTE=        2       ;TEMPS BEGIN ABOVE SAVED BP
0004                      13     DESSEG=         DESBYTE+2
0006                      14     DESRAST=        DESSEG+2
0008                      15     DESFLD=         DESRAST+2
000A                      16     DESX=           DESFLD+2
000C                      17     DESY=           DESX+2
000E                      18     WIDTH=          DESY+2
0010                      19     HEIGHT=         WIDTH+2
0012                      20     CLPX=           HEIGHT+2
0014                      21     CLPY=           CLPX+2
0016                      22     CLPWID=         CLPY+2
0018                      23     CLPHT=          CLPWID+2
001A                      24     SRCBYTE=        CLPHT+2
001C                      25     SRCSEG=         SRCBYTE+2
001E                      26     SRCRAST=        SRCSEG+2
0020                      27     SRCFLD=         SRCRAST+2
0022                      28     SRCX=           SRCFLD+2
0024                      29     SRCY=           SRCX+2
0026                      30     FUNC=           SRCY+2
0028                      31     GRAY=           FUNC+2
0014                      32     TABSIZ=         GRAY+2-DESBYTE/2        ;NO. WORDS IN CALLER'S TABLE
                          33     ;
                          34     ;       NOTE: PARALLEL TABLES  ...meaning that the source
                          35     ;       parmeters must lie the same distance from the destination
                          36     ;       parameters in both tables.  So check in the .LS file that
0000                      37             zero=   SRCBYTE-DESBYTE-SRCDEL+DESDEL
                          38     ;
                          39     ;       TEMP FRAME FORMAT
002A                      40     DESDEL= GRAY+2  ;4 VALS FOR Y MOD 4
0032                      41     DESWA=  DESDEL+8
0034                      42     STARTBITS=      DESWA+2
0036                      43     SKEW=   STARTBITS+2
0038                      44     SKMSK=  SKEW+2
003A                      45     HBAK=   SKMSK+2
003C                      46     VBAK=   HBAK+2
003E                      47     MASK1=  VBAK+2
0040                      48     MASK2=  MASK1+2
0042                      49     SRCDEL= MASK2+2 ;4 VALS FOR Y MOD 4
004A                      50     SRCWA=  SRCDEL+8
004C                      51     PRELD=  SRCWA+2
004E                      52     HCOUNT= PRELD+2
```

```
0050                    53    NWORDS= HCOUNT+2
0052                    54    SRCN=   NWORDS+2
0054                    55    SHIFTS= SRCN+2
0056                    56    OPN=    SHIFTS+2
0058                    57    GRAYS=  OPN+2   ;4 VALS FOR Y MOD 4
0060                    58    NTEMPS= GRAYS+8
                        59    ;
0008                    60    BRUSHFUNC=      8
000C                    61    CONSTFUNC=      0C              ;
                        62    ;
0218                    63            .LOC    .+1/2*2 ;EVENWORD ALIGNMENT
                        64    ;
                        65    ;       /* TABLE OF SOURCE ROUTINES
0218 07 05              66    SOURCES:        .ADR    SRC00P0
021A DD 04              67            .ADR    SRC0
021C DD 04              68            .ADR    SRC0
021E DD 04              69            .ADR    SRC0
0220 E3 04              70            .ADR    SRC1
0222 E3 04              71            .ADR    SRC1
0224 E3 04              72            .ADR    SRC1
0226 E3 04              73            .ADR    SRC1
0228 EB 04              74            .ADR    SRC2
022A EB 04              75            .ADR    SRC2
022C EB 04              76            .ADR    SRC2
022E EB 04              77            .ADR    SRC2
0230 03 05              78            .ADR    SRC3OP0
0232 FD 04              79            .ADR    SRC3
0234 FD 04              80            .ADR    SRC3
0236 FD 04              81            .ADR    SRC3
                        82    ;
                        83    ;       /* TABLE OF OPERATIONS
0238 17 05              84    OPS:    .ADR    OP0
023A 18 05              85            .ADR    OP1
023C 1C 05              86            .ADR    OP2
023E 20 05              87            .ADR    OP3
                        88    ;
                        89    ;       /* TABLE OF MASKS
0240 00 00              90    MASKTAB:        .ADR    0
0242 01 00              91            .ADR    1
0244 03 00              92            .ADR    3
0246 07 00              93            .ADR    7
0248 0F 00              94            .ADR    0F
024A 1F 00              95            .ADR    1F
024C 3F 00              96            .ADR    3F
024E 7F 00              97            .ADR    7F
0250 FF 00              98            .ADR    0FF
0252 FF 01              99            .ADR    1FF
0254 FF 03             100            .ADR    3FF
0256 FF 07             101            .ADR    7FF
0258 FF 0F             102            .ADR    0FFF
025A FF 1F             103            .ADR    1FFF
025C FF 3F             104            .ADR    3FFF
025E FF 7F             105            .ADR    7FFF
0260 FF FF             106            .ADR    0FFFF
                       107    ;
                       108    ;       /* TABLE OF SHIFTS (LEFT ROTATES)
0262 CA 05             109    SHIFTTABLE:     .ADR    SH0
0264 C8 05             110            .ADR    SH1
0266 C6 05             111            .ADR    SH2
0268 C4 05             112            .ADR    SH3
026A C2 05             113            .ADR    SH4
026C CB 05             114            .ADR    SH5
026E CD 05             115            .ADR    SH6
0270 CF 05             116            .ADR    SH7
0272 D1 05             117            .ADR    SH8
0274 D8 05             118            .ADR    SH9
0276 D6 05             119            .ADR    SH10
0278 D4 05             120            .ADR    SH11
027A DD 05             121            .ADR    SH12
027C DF 05             122            .ADR    SH13
027E E1 05             123            .ADR    SH14
0280 E3 05             124            .ADR    SH15
                       125    ;
0282 E9 52 02          126    JDONE:  JMP     DONE
                       127    ;
                       128    ;       /* ... and now, here's BITBLT! */
```

```
0285 83 EC 60            129    BITBLT: SUB     SP,#NTEMPS
0288 55                  130            PUSH    BP
0289 8B EC               131            MOV     BP,SP   ;NEW FRAME
028B 1E                  132            PUSH    DS      ;SAVE DS
                         133    ;
028C 8C D0               134            MOV     AX,SS   ;BLT CALLER'S TABLE TO LOCAL MEMORY
028E 8E C0               135            MOV     ES,AX
0290 8B FD               136            MOV     DI,BP
0292 83 C7 02            137            ADD     DI,#DESBYTE     ;STARTS AT OFFSET 2
0295 B9 14 00            138            MOV     CX,#TABSIZ
0298 FC                  139            CLD
0299 F2                  140            REP
029A A5                  141            MOVW
                         142    ;
                         143    ;       /* MAKE LOCAL COPY OF RECTANGLES, CHECKING BOUNDS
                         144    ;       SI offset =DESY-DESX for Y/HT, =0 for X/WID in this section:
029B BE 02 00            145            MOV     SI,#DESY-DESX   ;SI -> y and height
029E 8B 42 0A            146    YTHENX: MOV     AX,DESX!BP!SI
02A1 8B 4A 22            147            MOV     CX,SRCX!BP!SI
02A4 8B 52 0E            148            MOV     DX,WIDTH!BP!SI
02A7 8B 5A 12            149            MOV     BX,CLPX!BP!SI
02AA 3B D8               150            CMP     BX,AX   ;CHECK FOR DEST ORIGIN < CLIPPING
02AC 7E 13               151             JLE    XYOK
02AE 2B D8               152            SUB     BX,AX
02B0 03 CB               153            ADD     CX,BX   ;INCREASE SOURCE ORIGIN
02B2 2B D3               154            SUB     DX,BX   ;DECREASE EXTENT
02B4 03 C3               155            ADD     AX,BX   ;AND MOVE DEST ORIG TO CLIP ORIGIN
02B6 8B D8               156            MOV     BX,AX
02B8 89 42 0A            157            MOV     DESX!BP!SI,AX
02BB 89 4A 22            158            MOV     SRCX!BP!SI,CX
02BE 89 52 0E            159            MOV     WIDTH!BP!SI,DX
02C1 03 C2               160    XYOK:   ADD     AX,DX
02C3 03 5A 16            161            ADD     BX,CLPWID!BP!SI
02C6 2B D8               162            SUB     BX,AX   ;CHECK FOR DEST EXTENT TOO BIG
02C8 7D 05               163             JGE    WHOK
02CA 03 D3               164            ADD     DX,BX   ;REDUCE WIDTH THAT MUCH
02CC 89 52 0E            165            MOV     WIDTH!BP!SI,DX
02CF 85 D2               166    WHOK:   TEST    DX,DX
02D1 7E AF               167             JLE    JDONE   ;NO BLT IF WIDTH OR HEIGHT .LE. 0
02D3 83 EE 02            168            SUB     SI,#DESY-DESX   ;Now SI -> x and width
02D6 74 C6               169             JZ     YTHENX  ;BACK AGAIN TO DO THE X'S
                         170    ;
                         171    ;       /* CALCULATE SKEW FROM X'S
02D8 8B 46 22            172    FHX:    MOV     AX,SRCX!BP
02DB 8B 56 0A            173            MOV     DX,DESX!BP
02DE 2B C2               174            SUB     AX,DX
02E0 25 0F 00            175            AND     AX,#0F
02E3 89 46 36            176            MOV     SKEW!BP,AX      ;skew = (sourcex-destx) & 15
                         177    ;
                         178    ;       /* SET UP INCS AND MASKS ASSUMING NO OVERLAP
02E6 2B C0               179            SUB     AX,AX   ;zero
02E8 89 46 3A            180            MOV     HBAK!BP,AX      ;normal r to l (-1 means back)
02EB FC                  181            CLD                     ;same for hardware direction flag
02EC 89 46 3C            182            MOV     VBAK!BP,AX      ;normal top-to-bottom
02EF 8B C2               183            MOV     AX,DX
02F1 25 0F 00            184            AND     AX,#0F
02F4 BB 10 00            185            MOV     BX,#10
02F7 2B D8               186            SUB     BX,AX
02F9 89 5E 34            187            MOV     STARTBITS!BP,BX ;startbits← 16-(destx&15) [1-16]
02FC D1 E3               188            SHL     BX      ;WORD INDEX
02FE 2E                  189            SEG     CS
02FF 8B 87 40 02         190            MOV     AX,MASKTAB!BX
0303 89 46 3E            191            MOV     MASK1!BP,AX     ;mask1← MASKS!STARTBITS
0306 03 56 0E            192            ADD     DX,WIDTH!BP
0309 FF CA               193            DEC     DX
030B 81 E2 0F 00         194            AND     DX,#0F
030F BB 0F 00            195            MOV     BX,#0F
0312 2B DA               196            SUB     BX,DX
0314 D1 E3               197            SHL     BX      ;WORD INDEX
0316 2E                  198            SEG     CS
0317 8B 87 40 02         199            MOV     AX,MASKTAB!BX
031B F7 D0               200            NOT     AX
031D 89 46 40            201            MOV     MASK2!BP,AX     ;mask2 = NOT MASKS! (15-(destx+width-1)
**&15)
                         202    ;
                         203    ;       /* CHECK FOR POSSIBLE OVERLAP OF SOURCE AND DEST
```

```
0320 8B 46 04            204            MOV     AX,DESSEG!BP
0323 3B 46 1C            205            CMP     AX,SRCSEG!BP
0326 75 0B               206             JNE    NOLAP
0328 8B 46 02            207            MOV     AX,DESBYTE!BP
032B 3B 46 1A            208            CMP     AX,SRCBYTE!BP
032E 75 03               209             JNE    NOLAP
0330 E8 3A 02            210            CALL    OLAP
                         211    ;
                         212    ;       /* CALCULATE PRELD (NEED FOR EXTRA INITIAL WORD OF SOURCE)
0333 8B 4E 36            213    NOLAP:  MOV     CX,SKEW!BP
0336 85 C9               214            TEST    CX,CX
0338 74 16               215             JZ     SPRELD
033A 8B 5E 22            216            MOV     BX,SRCX!BP
033D 81 E3 0F 00         217            AND     BX,#0F
0341 2B CB               218            SUB     CX,BX
0343 FF C9               219            DEC     CX
0345 D1 C1               220            ROL     CX
0347 81 E1 01 00         221            AND     CX,#1   ;=0 if skew gr (sourcex&15) else 1
034B 03 4E 3A            222            ADD     CX,HBAK!BP      ;=0 if above XOR hbak=0
034E D1 E1               223            SHL     CX              ;else = +-2 dep on hbak
0350 89 4E 4C            224     SPRELD:        MOV     PRELD!BP,CX
                         225    ;
                         226    ;       /* SET UP SOURCES, OPS, AND GRAYS IF NECESSARY
0353 8B 4E 26            227            MOV     CX,FUNC!BP
0356 81 E1 0F 00         228            AND     CX,#0F
035A 89 4E 26            229            MOV     FUNC!BP,CX
035D 8B F1               230            MOV     SI,CX
035F D1 E6               231            SHL     SI      ;FUNC INDEXES WORD OF TABLE
0361 2E                  232            SEG     CS
0362 8B 84 18 02         233            MOV     AX,SOURCES!SI
0366 89 46 52            234            MOV     SRCN!BP,AX
0369 81 E6 06 00         235            AND     SI,#6   ;OP INDEXES OPS
036D 2E                  236            SEG     CS
036E 8B 84 38 02         237            MOV     AX,OPS!SI
0372 89 46 56            238            MOV     OPN!BP,AX
0375 83 F9 08            239            CMP     CX,#BRUSHFUNC
0378 7C 0D               240             JL     CNW     ;NO GRAY NEEDED
037A 83 F9 0C            241            CMP     CX,#CONSTFUNC
037D 7C 05               242             JL     GRAYP
037F C7 46 4C 00 00      243            MOV     PRELD!BP,##0    ;NO PRELOAD FOR CONST SOURCE
0384 E8 9F 01            244    GRAYP:  CALL    GRAYPREP        ;PREPARE 4-WORD TABLE OF GRAYS
                         245    ;
                         246    ;       /* CALC NWORDS (no. words stored), MERGE MASKS IF NEG
0387 8B 46 0E            247    CNW:    MOV     AX,WIDTH!BP
038A 2B 46 34            248            SUB     AX,STARTBITS!BP
038D FF C8               249            DEC     AX
038F 7D 0E               250             JGE    SNW
0391 8B 4E 40            251            MOV     CX,MASK2!BP     ;IF WIDTH<STARTBITS,
0394 21 4E 3E            252            AND     MASK1!BP,CX     ;THEN MASK1 = MASK1 .AND. MASK2
0397 C7 46 40 00 00      253            MOV     MASK2!BP,##0    ; AND MASK2 = 0
039C B8 F0 FF            254            MOV     AX,#-10         ; AND MAKE NWORDS=1
039F D1 F8               255    SNW:    SAR     AX
03A1 D1 F8               256            SAR     AX
03A3 D1 F8               257            SAR     AX
03A5 D1 F8               258            SAR     AX
03A7 05 02 00            259            ADD     AX,#2
03AA 89 46 50            260            MOV     NWORDS!BP,AX    ;NWORDS = 2+ (WIDTH-STARTBITS-1)/16
                         261    ;
                         262    ;       /* CALC CORE INCREMENTS BETWEEN SCAN LINES
                         263    ;       SI offset =SRC-DEST for SRC, =0 for DEST in this section:
                         264    ;       The operations XOR,SUB cause conditional NEG when
                         265    ;       used with the 0/-1 flags in HBAK,VBAK in this section
03AD BE 18 00            266            MOV     SI,#SRCRAST-DESRAST     ;SI -> src first
03B0 8B 46 4C            267            MOV     AX,PRELD!BP     ;0 OR +-2
03B3 25 02 00            268            AND     AX,#2   ;0 OR 2
03B6 D1 E8               269            SHR     AX      ;=1 IFF PRELOAD FOR SOURCE
03B8 03 46 50            270    STHEND: ADD     AX,NWORDS!BP    ;AX= NWORDS+(PRELOAD if source)
03BB 8B 56 3A            271            MOV     DX,HBAK!BP
03BE F7 D2               272            NOT     DX
03C0 33 C2               273            XOR     AX,DX
03C2 2B C2               274            SUB     AX,DX   ;(Negate NWORDS expr if left-to-right)
03C4 8B 4A 08            275            MOV     CX,DESFLD!BP!SI ;CX = FLD
03C7 8B 5A 06            276            MOV     BX,DESRAST!BP!SI
03CA 2B D9               277            SUB     BX,CX           ;BX = RAST-FLD
03CC 8B 56 3C            278            MOV     DX,VBAK!BP
03CF 33 CA               279            XOR     CX,DX
```

```
03D1 2B CA              280             SUB     CX,DX   ;(Negate both if bot-to-top)
03D3 33 DA              281             XOR     BX,DX
03D5 2B DA              282             SUB     BX,DX
03D7 03 D8              283             ADD     BX,AX   ;Now BX, CX = even, odd deltas
03D9 03 C8              284             ADD     CX,AX   ;  = +-FLD expr +-NWORDS expr
03DB 33 52 0C           285             XOR     DX,DESY!BP!SI   ;  PARITY OF VBAK, STARTING Y
03DE 33 56 10           286             XOR     DX,HEIGHT!BP    ;  AND STARTING VCOUNT (+1)
03E1 F7 C2 01 00        287             TEST    DX,#1           ;  DETERMINE
03E5 74 02              288              JZ     STDELS          ;  WHETHER TO SWITCH ODD, EVEN
03E7 87 D9              289             EXCHG   BX,CX
03E9 D1 E3              290     STDELS: SHL     BX      ;THESE DELTAS ARE IN WORDS
03EB D1 E1              291             SHL     CX
03ED 89 5A 2A           292             MOV     DESDEL+0!BP!SI,BX
03F0 89 4A 2C           293             MOV     DESDEL+2!BP!SI,CX
03F3 89 5A 2E           294             MOV     DESDEL+4!BP!SI,BX
03F6 89 4A 30           295             MOV     DESDEL+6!BP!SI,CX
                        296     ;
03F9 8B 42 0C           297             MOV     AX,DESY!BP!SI   ;/* CALC INITIAL CORE ADDRS
03FC 8B C8              298             MOV     CX,AX   ;STARTING Y
03FE 81 E1 01 00        299             AND     CX,#1
0402 F7 D9              300             NEG     CX
0404 23 4A 08           301             AND     CX,DESFLD!BP!SI ;CX← 0 IF EVEN, FLD IF ODD
0407 D1 F8              302             SAR     AX
0409 F6 62 06           303             MUL     DESRAST!BP!SI   ; Y/2 * RASTER
040C 03 C1              304             ADD     AX,CX   ; + FIELD LENGTH IF Y WAS ODD
040E 8B 52 0A           305             MOV     DX,DESX!BP!SI
0411 D1 FA              306             SAR     DX
0413 D1 FA              307             SAR     DX
0415 D1 FA              308             SAR     DX
0417 D1 FA              309             SAR     DX      ; PLUS STARTING X / 16
0419 03 C2              310             ADD     AX,DX
041B D1 E0              311             SHL     AX      ; WORD ADDRESS
041D 03 42 02           312             ADD     AX,DESBYTE!BP!SI
0420 89 42 32           313             MOV     DESWA!BP!SI,AX  ;STARTING MEMORY ADDRESS
0423 2B C0              314             SUB     AX,AX   ;NO PRELOAD FOR DEST
0425 83 EE 18           315             SUB     SI,#SRCRAST-DESRAST     ;SI now -> destination
0428 74 8E              316              JZ     STHEND          ;BACK TO DO DEST
                        317     ;
                        318     ;       SET UP SHIFT AND MASK FROM TABLES INDEXED BY SKEW
042A 8B 5E 36           319             MOV     BX,SKEW!BP      ;SKEW=  0,1,...,15
042D D1 E3              320             SHL     BX              ;WORD INDEX     (*2)
042F 2E                 321             SEG     CS
0430 8B 87 62 02        322             MOV     AX,SHIFTTABLE!BX        ;SKEW SHIFT DISPATCH
0434 89 46 54           323             MOV     SHIFTS!BP,AX
0437 F7 DB              324             NEG     BX
0439 81 E3 1E 00        325             AND     BX,#1E          ;       0,15,...,1 (*2)
043D 2E                 326             SEG     CS
043E 8B 87 40 02        327             MOV     AX,MASKTAB!BX
0442 74 03              328              JZ     SSKM            ;ZERO SKEW IGNORES DIRECTION
0444 33 46 3A           329             XOR     AX,HBAK!BP      ;COMP MASK IF BACKWARDS
0447 89 46 38           330     SSKM:   MOV     SKMSK!BP,AX
                        331     ;
044A 8E 5E 1C           332             MOV     DS,SRCSEG!BP    ;load seg regs
044D 8E 46 04           333             MOV     ES,DESSEG!BP
0450 8B 4E 4A           334             MOV     CX,SRCWA!BP     ;SOURCE MEMORY ADDRESS
0453 8B 7E 32           335             MOV     DI,DESWA!BP     ;DEST MEMORY ADDRESS
0456 8B 76 10           336             MOV     SI,HEIGHT!BP
0459 81 E6 03 00        337             AND     SI,#3
045D D1 E6              338             SHL     SI
045F EB 18              339              J      VL1
                        340     ;
                        341     ;
                        342     ;       /* VERTICAL LOOP FOR EACH SCAN-LINE */
                        343     ;
                        344     ;       /* SETUP
0461 8B CE              345     VLOOP:  MOV     CX,SI   ;STACHE CX WHILE SI INDEXES Y MOD 4
0463 8B 76 10           346             MOV     SI,HEIGHT!BP
0466 FF CE              347             DEC     SI
0468 7E 6D              348              JLE    DONE
046A 89 76 10           349             MOV     HEIGHT!BP,SI
046D 81 E6 03 00        350             AND     SI,#3   ;MOD 4
0471 D1 E6              351             SHL     SI      ;WORD INDEX
0473 03 7A 2A           352             ADD     DI,DESDEL!BP!SI
0476 03 4A 42           353             ADD     CX,SRCDEL!BP!SI
0479 8B 42 58           354     VL1:    MOV     AX,GRAYS!BP!SI
047C 89 46 28           355             MOV     GRAY!BP,AX
```

```
047F 8B F1              356             MOV     SI,CX
0481 8B 14              357             MOV     DX,0!SI
0483 03 76 4C           358             ADD     SI,PRELD!BP     ;0 OR +-2
0486 8B 5E 50           359             MOV     BX,NWORDS!BP
                        360     ;
                        361     ;       now DS.SI -> source, and ES.DI -> destination, and DF=dir
                        362     ;       AX = THE SOURCE DATA TO BE STORED
                        363     ;       BX = HORIZ WORD COUNT
                        364     ;       DX = WORD2 - THE OTHER SOURCE BITS FOR WORD-STRADDLING
                        365     ;
                        366     ;       /* INITIAL FRINGE
0489 FF 56 52           367             CALLI   SRCN!BP
048C 8B 4E 3E           368             MOV     CX,MASK1!BP
048F 23 C1              369             AND     AX,CX   ;MASK UNDER MASK1
0491 F7 D1              370             NOT     CX
0493 26                 371             SEG     ES
0494 23 0D              372             AND     CX,0!DI
0496 0B C1              373             OR      AX,CX
0498 AB                 374             STOW
0499 FF CB              375             DEC     BX
049B 74 C4              376              JZ     VLOOP
049D FF CB              377             DEC     BX
049F 74 24              378              JZ     HLAST
04A1 8B 4E 26           379             MOV     CX,FUNC!BP
04A4 83 F9 0C           380             CMP     CX,##CONSTFUNC
04A7 74 15              381              JE     BLKS
04A9 0B 4E 36           382             OR      CX,SKEW!BP
04AC 74 0A              383              JZ     BIT
                        384     ;
                        385     ;       /* FULL WORDS IN MIDDLE (IF ANY)
04AE FF 56 52           386     HLOOP:  CALLI   SRCN!BP
04B1 AB                 387             STOW            ;STORE UNMASKED
04B2 FF CB              388             DEC     BX
04B4 7F F8              389              JG     HLOOP
04B6 EB 0D              390              J      HLAST
                        391     ;
04B8 8B CB              392     BLT:    MOV     CX,BX   ;SPECIAL CASE OF UNSKEWED COPY
04BA F2                 393             REP             ; 8086 BLOCK TRANSFER
04BB A5                 394             MOVW
04BC EB 07              395              J      HLAST
                        396     ;
04BE 8B 46 28           397     BLKS:   MOV     AX,GRAY!BP      ;SPECIAL CASE OF STORE CONST
04C1 8B CB              398             MOV     CX,BX
04C3 F2                 399             REP             ; 8086 BLOCK STORE
04C4 AB                 400             STOW
                        401     ;
                        402     ;       /* FINAL FRINGE (IF ANY)
04C5 FF 56 52           403     HLAST:  CALLI   SRCN!BP
04C8 8B 4E 40           404             MOV     CX,MASK2!BP
04CB 23 C1              405             AND     AX,CX   ;MASK UNDER MASK2
04CD F7 D1              406             NOT     CX
04CF 26                 407             SEG     ES
04D0 23 0D              408             AND     CX,0!DI
04D2 0B C1              409             OR      AX,CX
04D4 AB                 410             STOW
04D5 EB 8A              411             J       VLOOP
                        412     ;
04D7 1F                 413     DONE:   POP     DS      ;RESTORE DATA SEGMENT
04D8 5D                 414             POP     BP
04D9 83 C4 60           415             ADD     SP,#NTEMPS
04DC C3                 416             RET
                        417     ;
                        418     ;
                        419     ;       /*HERE ARE THE FOUR SOURCES
04DD E8 27 00           420     SRC0:   CALL    SOURCE  ; /* WINDOW
04E0 FF 66 56           421             JMPI    OPN!BP
                        422     ;
04E3 E8 21 00           423     SRC1:   CALL    SOURCE  ; /* NOT WINDOW
04E6 F7 D0              424             NOT     AX
04E8 FF 66 56           425             JMPI    OPN!BP
                        426     ;
04EB E8 19 00           427     SRC2:   CALL    SOURCE  ; /* WINDOW .AND. GRAY
04EE 8B C8              428             MOV     CX,AX
04F0 F7 D1              429             NOT     CX
04F2 23 46 28           430             AND     AX,GRAY!BP      ;BLACK -> GRAY
04F5 26                 431             SEG     ES
```

```
04F6 23 0D              AND     CX,0!DI         ;WHITE -> TRANSPARENT
04F8 0B C1              OR      AX,CX
04FA FF 66 56           JMPI    OPN!BP
                        ;
04FD 8B 46 28   SRC3:   MOV     AX,GRAY!BP      ; /* CONSTANT (GRAY)
0500 FF 66 56           JMPI    OPN!BP
                        ;
0503 8B 46 28   SRC3OP0: MOV    AX,GRAY!BP      ; /* OPTIMIZED STORE CONST
0506 C3                 RET
                SRC0OP0:        ; /* OPTIMIZED MOVE (FALLS THRU SOURCE!)
                        ;
                        ;       SOURCE BITRECT
0507 8B 4E 38   SOURCE: MOV     CX,SKMSK!BP
050A 23 D1              AND     DX,CX   ;WORD2 .AND. SKEWMASK
050C F7 D1              NOT     CX
050E AD                 LODW
050F 23 C8              AND     CX,AX   ;NEXT WORD .AND. (NOT SKEWMASK)
0511 92                 EXCHG   AX,DX   ; (STASHES NEW WORD2 IN DX)
0512 0B C1              OR      AX,CX   ;COMBINE TWO PARTS INTO AX
0514 FF 66 54           JMPI    SHIFTS!BP       ;fast ROLV AX by SKEW!BP
                        ;
                        ;       MOV     CX,SKEW!BP
                        ;       ROLV    AX
                        ;       RET
                        ;
                        ;       /*HERE ARE THE FOUR OPERATIONS
0517 C3         OP0:    RET             ; /* SOURCE [MOVE]
                        ;
0518 26         OP1:    SEG     ES
0519 0B 05              OR      AX,0!DI ; /* SOURCE .OR. DEST [MERGE]
051B C3                 RET
                        ;
051C 26         OP2:    SEG     ES
051D 33 05              XOR     AX,0!DI ; /* SOURCE .XOR. DEST [XOR]
051F C3                 RET
                        ;
0520 F7 D0      OP3:    NOT     AX      ; /* (NOT SOURCE) .AND. DEST [ERASE]
0522 26                 SEG     ES
0523 23 05              AND     AX,0!DI
0525 C3                 RET
                        ;
                        ;
0526 8B 46 28   GRAYPREP:       MOV     AX,GRAY!BP      ;PICK UP THE GRAY WORD
0529 8B 56 3C           MOV     DX,VBAK!BP
052C 8B D8              MOV     BX,AX
052E 23 DA               AND    BX,DX           ;ONLY IF VBAK=-1 ...
0530 81 E3 0F 0F         AND    BX,##0F0F       ;WILL THESE 5 INSTRS
0534 32 FB               XOR    BH,BL           ;INVERT THE GRAY
0536 8A DF               MOV    BL,BH           ;FROM A-B-C-D
0538 33 C3               XOR    AX,BX           ;  TO A-D-C-B
053A 8B 4E 0C           MOV     CX,DESY!BP
053D 33 CA              XOR     CX,DX
053F 2B CA              SUB     CX,DX   ; +- STARTING Y, DEP ON VBAK
0541 03 4E 10           ADD     CX,HEIGHT!BP
0544 FF C1              INC     CX      ;FUDGE
0546 81 E1 03 00        AND     CX,#3   ; N ← (HEIGHT +- STARTING Y) MOD 4
054A D1 E1              SHL     CX
054C D1 E1              SHL     CX
054E D3 C0              ROLV    AX      ; ROT BY N NIBBLES TO ALIGN "SEAMS"
                ; NOW EXPAND EACH NIBBLE TO A FULL WORD AND STORE IN THE 4 GRAY WORDS
0550 BE 06 00           MOV     SI,#6
0553 B9 04 00           MOV     CX,#4
0556 8A F4      EXLP:   MOV     DH,AH   ; AX WAS A-B-C-D
0558 D3 C0              ROLV    AX
055A 8A D0              MOV     DL,AL
055C 81 E2 0F F0        AND     DX,##0F00F
0560 0A F2              OR      DH,DL
0562 8A D6              MOV     DL,DH   ; DX NOW A-A-A-A
0564 89 52 58           MOV     GRAYS!BP!SI,DX
0567 83 EE 02           SUB     SI,#2   ;ORDER IS A-B-C-D INTO 3-2-1-0
056A 7D EA               JGE    EXLP
056C C3                 RET
                        ;
                        ;       /* SPECIAL SETUP FOR HORIZ AND VERTICAL OVERLAP
056D 8B 46 0C   OLAP:   MOV     AX,DESY!BP
0570 8B 5E 24           MOV     BX,SRCY!BP
```

```
0573 3B C3              508             CMP     AX,BX
0575 7C 0C              509              JL     EASY    ;VERT DIR IS NORMAL AND NON-ZERO
0577 7F 34              510              JG     VOLAP   ;VERT IS BACKWARD
0579 8B 4E 0A           511             MOV     CX,DESX!BP      ;VDELTA=0, BETTER CHECK HDIR
057C 8B 56 22           512             MOV     DX,SRCX!BP
057F 3B CA              513             CMP     CX,DX
0581 7F 01              514              JG     HOLAP
0583 C3                 515     EASY:    RET                    ;HORIZ IS NORMAL
0584 C7 46 3A FF FF     516     HOLAP:  MOV     HBAK!BP,##-1    ;HORIZ BACKWARD (RIGHT-TO-LEFT)
0589 FD                 517             STD                     ;SAME FOR HARDWARE DIR
058A 8B 46 0E           518             MOV     AX,WIDTH!BP
058D FF C8              519             DEC     AX      ;WIDTH-1
058F 03 C8              520             ADD     CX,AX
0591 89 4E 0A           521             MOV     DESX!BP,CX      ;MOVE STARTING Xs TO RIGHT SIDE
0594 03 D0              522             ADD     DX,AX
0596 89 56 22           523             MOV     SRCX!BP,DX
0599 8B C1              524             MOV     AX,CX
059B 25 0F 00           525             AND     AX,#0F
059E FF C0              526             INC     AX
05A0 89 46 34           527             MOV     STARTBITS!BP,AX ;STARTBITS← (DESTX&15)+1 [1-16]
05A3 8B 46 3E           528             MOV     AX,MASK1!BP     ;EXCHANGE THE MASKS
05A6 87 46 40           529             EXCHG   AX,MASK2!BP
05A9 89 46 3E           530             MOV     MASK1!BP,AX
05AC C3                 531              RET
                        532     ;
05AD C7 46 3C FF FF     533     VOLAP:  MOV     VBAK!BP,##-1    ;VERT BACKWARD (BOTTOM-TO-TOP)
05B2 8B 4E 10           534             MOV     CX,HEIGHT!BP
05B5 FF C9              535             DEC     CX
05B7 03 C1              536             ADD     AX,CX
05B9 89 46 0C           537             MOV     DESY!BP,AX      ;MOVE STARTING Ys TO BOTTOM
05BC 03 D9              538             ADD     BX,CX
05BE 89 5E 24           539             MOV     SRCY!BP,BX
05C1 C3                 540             RET
                        541     ;
                        542     ;       /* OPTIMUM 8086 SHIFT ROUTINES:
05C2 D1 C0              543     SH4:    ROL     AX      ;SHIFTS 0-4
05C4 D1 C0              544     SH3:    ROL     AX
05C6 D1 C0              545     SH2:    ROL     AX
05C8 D1 C0              546     SH1:    ROL     AX
05CA C3                 547     SH0:    RET
                        548     ;
05CB D1 C8              549     SH5:    ROR     AX      ;SHIFTS 5-8
05CD D1 C8              550     SH6:    ROR     AX
05CF D1 C8              551     SH7:    ROR     AX
05D1 86                 552     SH8:    86      ;**     HAND ASSEMBLY OF:
05D2 C4                 553             0C4     ;**     EXCHG   AH,AL
05D3 C3                 554             RET
                        555     ;
05D4 D1 C0              556     SH11:   ROL     AX      ;SHIFTS 9-11
05D6 D1 C0              557     SH10:   ROL     AX
05D8 D1 C0              558     SH9:    ROL     AX
05DA 86                 559             86      ;**     HAND ASSEMBLY OF:
05DB C4                 560             0C4     ;**     EXCHG   AH,AL
05DC C3                 561             RET
                        562     ;
05DD D1 C8              563     SH12:   ROR     AX      ;SHIFTS 12-15
05DF D1 C8              564     SH13:   ROR     AX
05E1 D1 C8              565     SH14:   ROR     AX
05E3 D1 C8              566     SH15:   ROR     AX
05E5 C3                 567             RET
                        568
                         24
                         25             .Get    "Z-Subs.bca"            ;misc. subroutines
                          1     ;       Z-Subs.bca
                          2
                          3     ;       This file contains miscellaneous subroutines
                          4     ;       to be called from Smalltalk (in the interim)
                          5     ;       or from other routines resident in rom.
                          6     ;       Author: Bruce Horn
                          7     ;       Last changed: May 7, 1979  12:32 AM
                          8
                          9     ;       BltOld, BltNew are in DoCursor.bca
                         10
004A                     11     ukbrq=          4A      ;request keyboard data
0160                     12     sa=             160     ;starting address port, CRT
                         13
```

```
                        14    Retrace:
05E6 A1 36 21           15            mov     ax,DisplayBits  ;get byte address and segment of
05E9 8B 1E 38 21        16            mov     bx,DisplayBits+2        ;bitmap/cursor table
05ED D1 E8              17            shr     ax
05EF D1 E8              18            shr     ax
05F1 D1 E8              19            shr     ax              ;divide byte adr by 8
05F3 D1 E3              20            shl     bx              ;multiply segment by 2
05F5 03 C3              21            add     ax,bx           ;then add the terms->19bits/3
05F7 BA 60 01           22            mov     dx,#sa          ; starting address port
05FA EF                 23            outdw
05FB C3                 24            ret                     ;return to caller
                        25
                        26    WaitASecond:
05FC B9 05 00           27            mov     cx,#5
05FF B8 F0 7F           28            mov     ax,#7FF0
                        29    LP1:
0602 FF C8              30            dec     ax
0604 75 FC              31            jnz     LP1
0606 B8 F0 7F           32            mov     ax,#7FF0
0609 E2 F7              33            loop    LP1
060B C3                 34            ret
                        35
                        36
                        37    IntsOff:
060C BA 02 00           38            mov     dx,#ocw1        ;get old mask
060F EC                 39            ind
0610 2E                 40            seg     cs
0611 A3 0A 10           41            mov     IRR,ax          ;save whole word (only al sig)
0614 B0 FF              42            mov     al,#0FF         ;mask off all interrupts
0616 EE                 43            outd
0617 C3                 44            ret
                        45
                        46    IntsOn:
0618 2E                 47            seg     cs
0619 A1 0A 10           48            mov     ax,IRR          ;get old interrupt mask
061C BA 02 00           49            mov     dx,#ocw1
061F EE                 50            outd                    ;send al out to port 2 and
0620 C3                 51            ret                     ;restore interrupts
                        52
                        53
                        54    DisplayOff:
0621 B8 00 00           55            mov     ax,#0
0624 BA 54 01           56            mov     dx,#154         ;reset display controller
0627 EE                 57            outd
0628 C3                 58            ret
                        59
                        60    DisplayOn:
0629 B8 00 00           61            mov     ax,#0
062C BA 5C 01           62            mov     dx,#15C
062F EE                 63            outd                    ;start CRT timing chain
0630 C3                 64            ret
                        65
                        66    WaitBug:
0631 2E                 67            seg     cs
0632 A1 18 20           68            mov     ax,KbdStatus    ;get keyboard status
0635 3C 00              69            cmp     al,#0           ;are mouse buttons zero?
0637 74 F8              70            je      WaitBug         ;yes, try again, otherwise
0639 C3                 71            ret                     ;return to caller
                        72
                        73    WaitNoBug:
063A 2E                 74            seg     cs
063B A1 18 20           75            mov     ax,KbdStatus
063E 3C 00              76            cmp     al,#0
0640 75 F8              77            jne     WaitNoBug       ;If any buttons down, loop
0642 C3                 78            ret
                        79
                        80    ShowNewCursor:
0643 2E                 81            seg     cs
0644 A3 20 10           82            mov     R0,ax           ;save source bitmap
0647 E8 63 00           83            call    HideCursor      ;hide old cursor
064A 2E                 84            seg     cs
064B A1 20 10           85            mov     ax,R0           ;restore source bitmap
064E 2E                 86            seg     cs
064F A3 4E 21           87            mov     CursorBits,ax   ;new source byte address
0652 E8 6B 00           88            call    ShowCursor      ;show new cursor
0655 C3                 89            ret
```

```
                        90
                        91    SetP100:
0656 B9 00 00           92            mov     cx,#0
0659 EB 16              93            j       SetPort
                        94    SetP60:
065B B9 01 00           95            mov     cx,#1
065E EB 11              96            j       setPort
                        97    SetP20:
0660 B9 02 00           98            mov     cx,#2
0663 EB 0C              99            j       SetPort
                       100
                       101    Ports:
0665 0C 10             102            .adr    Port100
0667 00 01             103            .adr    100
0669 0E 10             104            .adr    Port60
066B 60 00             105            .adr    60
066D 10 10             106            .adr    Port20
066F 20 00             107            .adr    20
                       108
                       109
                       110    ;ax=bits (right justified), bx=field spec, cx=port offset
                       111    SetPort:
0671 D1 E1             112            shl     cx
0673 D1 E1             113            shl     cx              ;point to correct place in table
0675 8B F1             114            mov     si,cx           ;offset in table
0677 8B CB             115            mov     cx,bx           ;move field definition into cx
0679 2E                116            seg     cs
067A 8B BC 65 06       117            mov     di,Ports!si     ;di is address of Port100/60/20
067E 2E                118            seg     cs
067F 23 1D             119            and     bx,0!di         ;and    bx, Port100/60/20
0681 D1 E9             120    NS:     shr     cx              ;shift field def by 1
0683 73 04             121            jnc     SD              ;shift done
0685 D1 E0             122            shl     ax              ;shift bits left 1
0687 EB F8             123            j       NS
0689 0B C3             124    SD:     or      ax,bx           ;or in new bits
068B 2E                125            seg     cs
068C 89 05             126            mov     0!di,ax         ;save new Port100/60/20
068E FF C6             127            inc     si
0690 FF C6             128            inc     si              ;next number in table
0692 2E                129            seg     cs
0693 8B 94 65 06       130            mov     dx,Ports!si
0697 EF                131            outdw
0698 C3                132            ret
                       133
                       134
                       135    ;DS,SI point to table for initialization of display
                       136
                       137    initDC:                         ;initialize display controller
0699 8B 0C             138            mov     cx,0!si         ;length of table is first word
069B FF C6             139            inc     si
069D FF C6             140            inc     si
069F 8B 14             141    Param:  mov     dx,0!si         ;get port of store
06A1 FF C6             142            inc     si
06A3 FF C6             143            inc     si
06A5 8A 04             144            mov     al,0!si         ;and get value
06A7 EE                145            outd                    ;send al out to port, then
06A8 FF C6             146            inc     si              ;increment si for next word
06AA E2 F3             147            loop    Param           ;and get next
06AC C3                148            ret                     ;return
                       149
                       150    ;blts out, locks cursor
                       151
                       152    HideCursor:
06AD B8 FF FF          153            mov     ax,#0FFFF
06B0 2E                154            seg     cs
06B1 3B 06 62 21       155            cmp     ax,CursorLock   ;is it already locked?
06B5 75 01             156            jne     HideIt
06B7 C3                157            ret                     ;already hidden
                       158    HideIt:
06B8 2E                159            seg     cs
06B9 A3 62 21          160            mov     CursorLock,ax   ;lock it
06BC E8 16 03          161            call    BltOld          ;Blt out old cursor
06BF C3                162            ret
                       163
                       164    ;blts in, unlocks new cursor
                       165
```

```
                        166     ShowCursor:
06C0 B8 00 00           167             mov     ax,#0
06C3 2E                 168             seg     cs
06C4 3B 06 62 21        169             cmp     ax,CursorLock   ;is it already shown?
06C8 75 01              170             jne     ShowIt          ;keep cursor locked
06CA C3                 171             ret                     ;already shown
                        172     ShowIt:
06CB E8 10 03           173             call    BltNew          ;Blt in new cursor
06CE 2E                 174             seg     cs
06CF C7 06 62 21 00 00  175             mov     CursorLock,##0  ;allow further cursor moves
06D5 C3                 176             ret
                        177
                        178     ;begins kbd byte transfer through uart
                        179
                        180     RdKbd:
06D6 B0 00              181             mov     al,#0           ;Request keyboard data
06D8 E6 4A              182             out     ukbrq
06DA C7 06 12 10 00 00  183             mov     KbdOffset,##0   ;and set offset to zero
06E0 C3                 184             ret                     ;let interrupt handle bytes
                        185
                        186             .END
                         26
                         27             .Get    "Z-IOInit.bca"          ;interrupt inits
                          1     ;       Z-IOInit.bca
                          2
                          3     ;       This file contains the I/O initialization
                          4     ;       code for the display and kbd
                          5
                          6     ;       Author: Bruce Horn
                          7     ;       Last changed: May 2, 1979  9:00 AM
                          8
                          9
004E                      10    ureset=         4E      ;reset UART
0016                      11    ucbits=         16      ;proper bits for control reg
0048                      12    uctrl=          48      ;control location
0001                      13    TCGo=           1       ;Timing chain on
0005                      14    CRT12Mhz=       5       ;Bit clock @ 12 Mhz
                          15
                          16
                          17
                          18    DCParams:
06E1 09 00                19            .adr    9.      ;number of parameters in table
06E3 40 01                20            .adr    140
06E5 5F                   21                    05F     ;0: CharTimes/Horizontal period
06E6 42 01                22            .adr    142
06E8 F2                   23                    0F2     ;1: 1=Interlace bit;CharTimes/HorizSync
06E9 44 01                24            .adr    144
06EB 7D                   25                    07D     ;2: 80 chars/line;15+1 scans/char
06EC 46 01                26            .adr    146
06EE 1D                   27                    01D     ;3: 00 skew; 30 dataRows/field-1
06EF 48 01                28            .adr    148
06F1 04                   29                    004     ;4: 521 rasterScans/field -513/2
06F2 4A 01                30            .adr    14A
06F4 10                   31                    010     ;5: delay from VSynch to first data row
06F5 54 01                32            .adr    154
06F7 00                   33                    0       ;Reset chip
06F8 5A 01                34            .adr    15A
06FA 1E                   35                    01E     ;13: cursor line 30
06FB 5C 01                36            .adr    15C
06FD 00                   37                    0       ;Start timing chain
                          38
                          39
                          40    IOInit:
06FE FC                   41            cld                     ;clear direction
                          42    ;set up 8259 for interrupts, but don't enable interrupts at all
                          43
06FF BA 00 00             44            mov     dx,#ic          ;ICW1
0702 B8 13 00             45            mov     ax,#13          ;single, ic4
0705 EE                   46            outd
0706 BA 02 00             47            mov     dx,#ic+2        ;ICW2
0709 B8 20 00             48            mov     ax,#20          ;vector begins at 20*2 = 80
070C EE                   49            outd
070D B8 0D 00             50            mov     ax,#0D          ;ICW4, is an 8086, buffered mode
0710 EE                   51            outd
0711 B8 FF 00             52            mov     ax,#0FF         ;mask should be all disabled
0714 EE                   53            outd
```

```
0715 B8 00 00            54          mov     ax #0           ;
0718 E7 4E               55          outw    ureset          ;reset the uart on kbd
071A B8 16 00            56          mov     ax, #ucbits
071D E7 48               57          outw    uctrl           ;load ctrl reg
                         58
071F BE E1 06            59          mov     si,#DCParams    ;table of init parameters
0722 E8 74 FF            60          call    initDC          ;Initialize display controller
0725 B8 05 00            61          mov     ax,#CRT12Mhz    ;set port 100 for 12mhz clock
0728 BB FF F1            62          mov     bx,#BitClk      ;field specification
072B E8 28 FF            63          call    SetP100
072E B8 01 00            64          mov     ax,#TCGo        ;timing chain begin
0731 BB FF DF            65          mov     bx,#GoWordClock ;field specification
0734 E8 1F FF            66          call    SetP100         ;and set port 100
                         67
0737 BA 02 00            68          mov     dx,#ocw1
073A EC                  69          ind                     ;get the current IRR
073B 24 EF               70          and     al,#ir4         ;enable OddInt
073D 24 BF               71          and     al,#ir6         ;and KbdInt
073F EE                  72          outd                    ;send it outd to ocw1
0740 2E                  73          seg     cs
0741 A3 0A 10            74          mov     IRR,ax          ;save mask in IRR
0744 FB                  75          sti                     ;enable interrupts
0745 C3                  76          ret                     ; return to debugger
                         77
                         78
                         79      ;Calls SetP100, InitDC (subs.bca)
                         80
                         81              .END
                         28
                         29              .Get    "Z-FieldInts.bca"               ;fielding interrupts
                          1      ;       Z-FieldInts.bca
                          2
                          3      ;       This file contains the calls on the interrupt
                          4      ;       routines.
                          5
                          6      ;       Author: Bruce Horn
                          7      ;       Last changed: May 2, 1979  2:52 PM
                          8
0020                      9      eoi=    20              ;end of interrupt
0042                     10      udatin= 42              ;obtain keyboard data
004C                     11      urrda=  4C              ;Reset data ready on UART
                         12
                         13      OddInt:
0746 50                  14              push    ax              ;14!sp
0747 52                  15              push    dx              ;12!sp
0748 53                  16              push    bx              ;10!sp
0749 51                  17              push    cx              ;8!sp
074A 56                  18              push    si              ;6!sp
074B 57                  19              push    di              ;4!sp
074C 1E                  20              push    ds              ;2!sp
074D 06                  21              push    es              ;0!sp
                         22
074E B8 00 00            23              mov     ax,#CurrentDS
0751 8E D8               24              mov     ds,ax
0753 8E C0               25              mov     es,ax
0755 FF 06 14 20         26              incw    ICount          ;increment interrupt count
0759 E8 8A FE            27              call    Retrace         ;do vertical retrace for crt
075C E8 D8 00            28              call    EncodeKbd       ;Interpret kbd bytes
075F E8 48 02            29              call    DoCursor        ;update cursor
0762 E8 71 FF            30              call    RdKbd           ;start fetching new kbd bytes
0765 07                  31              pop     es              ;pop all registers
0766 1F                  32              pop     ds              ;except ax, dx for interrupt
0767 5F                  33              pop     di              ;dismissal
0768 5E                  34              pop     si
0769 59                  35              pop     cx
076A 5B                  36              pop     bx
                         37
076B B0 20               38              mov     al,#eoi         ;dismiss interrupt
076D BA 00 00            39              mov     dx,#ic          ;send to 8259
0770 FA                  40              cli                     ;DON'T service interrupts...
0771 EE                  41              outd
                         42
0772 5A                  43              pop     dx
0773 58                  44              pop     ax
0774 FB                  45              sti                     ;...until now---all safe
0775 CF                  46              iret                    ;pop flags on return, also cs,ip
```

```
                        47
                        48
                        49    KbdInt:
0776 50                 50            push    ax
0777 53                 51            push    bx
0778 52                 52            push    dx
0779 1E                 53            push    ds
077A B8 00 00           54            mov     ax,#CurrentDS
077D 8E D8              55            mov     ds,ax
077F 8B 1E 12 10        56            mov     bx,KbdOffset
0783 B8 0C 00           57            mov     ax,#12.         ;if greater than 12, don't store
0786 3B D8              58            cmp     bx,ax           ;next byte, otherwise its
0788 7D 10              59            jge     Fini            ;ok for next kbd byte
078A E4 42              60            in      udatin          ;get data
078C 88 87 2A 20        61            mov     KbdData!bx,al   ;low byte is data
0790 FF C3              62            inc     bx
0792 89 1E 12 10        63            mov     KbdOffset,bx
0796 B0 00              64            mov     al,#0           ;store it
0798 E6 4C              65            out     urrda           ;Reset DR
079A 1F                 66    Fini:   pop     ds              ;begin restoring registers
079B B0 20              67            mov     al,#eoi         ;tell 8259 end of interrupt
079D BA 00 00           68            mov     dx,#ic
07A0 FA                 69            cli                     ;Don't service interrupts...
07A1 EE                 70            outd
07A2 5A                 71            pop     dx
07A3 5B                 72            pop     bx
07A4 58                 73            pop     ax
07A5 FB                 74            sti                     ;until now--all safe
07A6 CF                 75            iret                    ;interrupt return
                        76
                        77
                        78    ParErrInt:
07A7 B8 06 00           79            mov     ax,#6
07AA BB F0 FF           80            mov     bx,#LedsOn
07AD E8 B0 FE           81            call    SetP20
07B0 E8 65 FE           82            call    IntsOn
07B3 E8 73 FE           83            call    DisplayOn
07B6 2E                 84            seg     cs
07B7 A1 A9 0D           85            mov     ax,.ErrorCursor
07BA E8 86 FE           86            call    ShowNewCursor
07BD EB FE              87    I0:     j       I0              ;stop machine
                        88
                        89    IPSysInt:
07BF B8 07 00           90            mov     ax,#7
07C2 BB F0 FF           91            mov     bx,#LedsOn
07C5 E8 98 FE           92            call    SetP20
07C8 E8 4D FE           93            call    IntsOn
07CB E8 5B FE           94            call    DisplayOn
07CE 2E                 95            seg     cs
07CF A1 A9 0D           96            mov     ax,.ErrorCursor
07D2 E8 6E FE           97            call    ShowNewCursor
07D5 EB FE              98    I1:     j       I1              ;stop machine
                        99
                       100    DiskInt:
07D7 B8 08 00          101            mov     ax,#8
07DA BB F0 FF          102            mov     bx,#LedsOn
07DD E8 80 FE          103            call    SetP20
07E0 E8 35 FE          104            call    IntsOn
07E3 E8 43 FE          105            call    DisplayOn
07E6 2E                106            seg     cs
07E7 A1 A9 0D          107            mov     ax,.ErrorCursor
07EA E8 56 FE          108            call    ShowNewCursor
07ED EB FE             109    I2:     j       I2              ;stop machine
                       110
                       111    EIAInt:
07EF B8 09 00          112            mov     ax,#9
07F2 BB F0 FF          113            mov     bx,#LedsOn
07F5 E8 68 FE          114            call    SetP20
07F8 E8 1D FE          115            call    IntsOn
07FB E8 2B FE          116            call    DisplayOn
07FE 2E                117            seg     cs
07FF A1 A9 0D          118            mov     ax,.ErrorCursor
0802 E8 3E FE          119            call    ShowNewCursor
0805 EB FE             120    I3:     j       I3
                       121
                       122    ADCInt:
```

```
0807 B8 0A 00           123             mov     ax,#0A
080A BB F0 FF           124             mov     bx,#LedsOn
080D E8 50 FE           125             call    SetP20
0810 E8 05 FE           126             call    IntsOn
0813 E8 13 FE           127             call    DisplayOn
0816 2E                 128             seg     cs
0817 A1 A9 0D           129             mov     ax,.ErrorCursor
081A E8 26 FE           130             call    ShowNewCursor
081D EB FE              131     I4:     j       I4
                        132
                        133     VSyncInt:
081F B8 0B 00           134             mov     ax,#0B
0822 BB F0 FF           135             mov     bx,#LedsOn
0825 E8 F0 FD           136             call    IntsOn
0828 E8 FE FD           137             call    DisplayOn
082B E8 32 FE           138             call    SetP20
082E 2E                 139             seg     cs
082F A1 A9 0D           140             mov     ax,.ErrorCursor
0832 E8 0E FE           141             call    ShowNewCursor
0835 EB FE              142     I5:     j       I5
                        143
                        144
                        145             .END
                         30
                         31             .Get    "Z-EncodeKbd.bca"               ;Encoding keyboard
                          1     ;       Z-EncodeKbd.bca
                          2
                          3     ;       This file contains the keyboard encoder and
                          4     ;       interrupt key handler
                          5     ;       Author: Bruce Horn
                          6     ;       Last changed: April 30, 1979  3:11 PM
                          7
                          8
                          9     ;Other useful constants
0003                      10    WordZero=       3.      ;Beginning of actual keyboard words
                         11
                         12
                         13     ;The following are META key bit locations--STRANGE things happen
                         14     ;to those who fool with META keys--
2F7F                     15     ZeroMeta0=      02F7F   ;mask off ms1, rsh,lock and ms3
FB6F                     16     ZeroMeta1=      0FB6F   ;mask off ctrl, ms2,  and lsh
D080                     17     OneMeta0=       0D080   ;mask on...
0490                     18     OneMeta1=       00490   ;mask on...
8000                     19     MetaMB1=        8000    ;in word 0
0080                     20     MetaMB2=        80      ;in word 1
0080                     21     MetaMB3=        80      ;in word 0
0010                     22     MetaLS=         10      ;in word 1
4000                     23     MetaRS=         4000    ;in word 0
0400                     24     MetaC=          400     ;in word 1
1000                     25     MetaLock=       1000    ;in word 0
0040                     26     OSShift=        64.     ;offset for shift key down, and for
00C0                     27     OSCTRL=         192.    ;control down--add for control+shift
0080                     28     OSLock=         128.    ;offset for lock down ONLY
0020                     29     CBLength=       32.     ;length of circular buffer, in words
                         30
                         31
                         32     ;The following uart/read loop will ultimately be interrupt-driven
                         33     ;by the uart itself.  Currently it tests the uart status, and if not re
**ady, it runs in a wait loop.
                         34
                         35     EncodeKbd:
0837 C7 06 18 20 00 00   36             mov     KbdStatus,##0   ;clear mouse buttons at beginning
083D A0 2C 20            37             mov     al,MXDelta      ;get x delta from read
0840 98                  38             cbw                     ;for add, convert to word
0841 01 06 1A 20         39             add     MouseX,ax       ;and add delta
0845 A0 2B 20            40             mov     al,MYDelta      ;get mouse Y byte
0848 98                  41             cbw                     ;for sub, convert to word
0849 29 06 1C 20         42             sub     MouseY,ax       ;and sub delta (y backwards!)
084D BB 02 00            43             mov     bx,#2           ;do bounds checking on mouse
                         44     YTX:
0850 8B 87 1A 20         45             mov     ax,MouseX!bx    ;first MouseY then MouseX
0854 8B 8F 22 20         46             mov     cx,RMW!bx       ;first RMH then RMW
0858 8B 97 1E 20         47             mov     dx,RMX!bx       ;first RMY then RMX
085C 03 CA               48             add     cx,dx           ;for corner
085E 3B C1               49             cmp     ax,cx           ;is MouseY > ClipH?
0860 7F 0B               50             jg      OOBC            ;yes, Out of Bounds CX
```

```
0862 3B C2                51          cmp     ax,dx           ;is MouseY < ClipY?
0864 7C 0D                52          jl      OOBD            ;yes, Out Of Bounds DX
                          53  N:
0866 83 EB 02             54          sub     bx,#2           ;offset zero, now
0869 74 E5                55          jz      YTX             ;go do X's
086B EB 0C                56          j       FIB             ;Forced In Bounds
                          57  OOBC:
086D 89 8F 1A 20          58          mov     MouseX!bx,cx    ;force upper limit in MouseY/X
0871 EB F3                59          j       N
                          60  OOBD:
0873 89 97 1A 20          61          mov     MouseX!bx,dx    ;force lower limit in MouseY/X
0877 EB ED                62          j       N
                          63
                          64  FIB:                            ;forced in bounds
0879 BA 00 00             65          mov     dx,#0           ;0 means no META keys yet
087C C7 06 06 10 00 00    66          mov     Offset,##0      ;offset starts at 0
0882 A1 2D 20             67          mov     ax,KW0          ;ax=KbdData word 0
0885 8B 1E 2F 20          68          mov     bx,KW1          ;bx=KbdData word 1
                          69
0889 B9 80 D0             70          mov     cx,#OneMeta0
088C 85 C1                71          test    ax,cx           ;are any META keys down in W0?
088E 74 21                72          jz      MetaW1          ;No, try word 1
0890 A9 00 80             73          test    ax,#MetaMB1     ;is it mouse button 1?
0893 74 05                74          jz      M1              ;no, try next
0895 80 0E 18 20 04       75          or      KbdStatus,#4    ;turn the 4 bit on
                          76  M1:
089A A9 00 40             77          test    ax,#MetaRS      ;is it right shift?
089D 74 08                78          jz      M2              ;no...
089F FF C2                79          inc     dx              ;say META key down
08A1 C7 06 06 10 40 00    80          mov     Offset,##OSShift        ;make offset a shift
                          81  M2:
08A7 A9 80 00             82          test    ax,#MetaMB3     ;is mouse button 3 down?
08AA 74 05                83          jz      MetaW1          ;no, try next
08AC 80 0E 18 20 01       84          or      KbdStatus,#1    ;and a 1 in the status word
                          85  MetaW1:
08B1 B9 90 04             86          mov     cx,#OneMeta1
08B4 85 D9                87          test    bx,cx           ;any META keys down in this word?
08B6 74 23                88          jz      TryLock         ;nope, go to end
08B8 F7 C3 10 00          89          test    bx,#MetaLS      ;is it left shift?
08BC 74 08                90          jz      M3              ;no, try control and MS2
08BE FF C2                91          inc     dx              ;inc dx for shift flag
08C0 C7 06 06 10 40 00    92          mov     Offset,##OSShift        ;put shift offset in loc
                          93  M3:
08C6 F7 C3 00 04          94          test    bx,#MetaC       ;is it control?
08CA 74 0A                95          jz      M4              ;if it isn't, it's mouse button 2
08CC FF C2                96          inc     dx              ;META key in effect
08CE 81 06 06 10 C0 00    97          add     Offset,##OSCTRL ;add control offset
08D4 EB 05                98          j       TryLock         ;now see if we can do ShiftLock
08D6 80 0E 18 20 02       99  M4:     or      KbdStatus,#2    ; MS2, and the two bit in
                         100
                         101  TryLock:
08DB 83 FA 00            102          cmp     dx,#0           ;is any other META key down?
08DE 75 0B               103          jne     EndMetaKeys     ;Yes, ignore if Lock down
08E0 A9 00 10            104          test    ax,#MetaLock    ;is Lock down by itself?
08E3 74 06               105          jz      EndMetaKeys     ;no, overridden by other METAs
08E5 C7 06 06 10 80 00   106          mov     Offset,##OSLock ;yes, put lock offset in location
                         107
                         108  EndMetaKeys:
08EB 25 7F 2F            109          and     ax,#ZeroMeta0   ;zero the meta bits
08EE 81 E3 6F FB         110          and     bx,#ZeroMeta1   ;in both words
08F2 89 1E 2F 20         111          mov     KW1,bx          ;put it back
08F6 A3 2D 20            112          mov     KW0,ax          ;both words
08F9 C7 06 02 10 04 00   113          mov     Count,##4       ;loop 4 times
08FF C7 06 00 10 00 00   114          mov     BitsGone,##0    ;start with zeroth bit
0905 BE 03 00            115          mov     si,#WordZero    ;and word zero
                         116  NextWd:
0908 8B 84 14 10         117          mov     ax,OldKbdData!si        ;get the data from UART fetch
090C 8B 9C 2A 20         118          mov     bx,KbdData!si   ;get old data from last int.
0910 33 C3               119          xor     ax,bx           ;xor the words, then land in new,
0912 23 C3               120          and     ax,bx           ;giving 1's=key w/down transition
0914 75 15               121          jnz     ProcessWord     ;if not zero, process word
                         122
                         123  NewSetOfBits:
0916 89 9C 14 10         124          mov     OldKbdData!si,bx        ;save kbd word in old table
091A 83 06 00 10 10      125          add     BitsGone,##10   ;add 10H for next word
091F FF C6               126          inc     si
```

```
0921 FF C6                      127             inc     si              ;increment index reg
0923 FF 0E 02 10                128             decw    Count           ;decrement loop count
0927 75 DF                      129             jnz     NextWd          ;and loop back
0929 EB 6F                      130             j       EndKbd          ;End keyboard handler
                                131
                                132     ProcessWord:
092B C7 06 04 10 01 00          133             mov     Mask,##1        ;one bit on
0931 B9 10 00                   134             mov     cx,#10          ;start with low order bit
                                135     NextBit:
0934 85 06 04 10                136             test    ax,Mask         ;is this bit a 1?
0938 74 57                      137             jz      NewBit          ;no, get the next bit
093A 8B 16 00 10                138             mov     dx,BitsGone     ;get word*16 as bits
093E 03 D1                      139             add     dx,cx           ;add this bit number,-->index
0940 FF CA                      140             dec     dx              ;but it's off by one
0942 03 16 06 10                141             add     dx,Offset       ;add the offset
0946 8B 3E 0C 21                142             mov     di,NIChars      ;how many interrupt chars
094A 83 FF 00                   143             cmp     di,#0           ;are there any interrupts?
094D EB 22                      144             j       EndIC           ;no, go to end (was je EndIc)
094F 89 3E 08 10                145             mov     Temp1,di        ;save it in Temp1
0953 BF 00 00                   146             mov     di,#0           ;start with zero offset
                                147
                                148     NC:
0956 3B 95 0E 21                149             cmp     dx,ICList!di    ;compare with first character
095A 75 09                      150             jne     NotThisOne      ;not this character
095C FF C7                      151             inc     di              ;zero=no chars, so add 1
095E 89 3E 7A 20                152             mov     IntKey,di       ;save it in flag register
0962 E9 2C 00                   153             jmp     NewBit          ;only 1 interrupt at a time
                                154
                                155     NotThisOne:
0965 FF 0E 08 10                156             decw    Temp1           ;decrement count
0969 74 06                      157             jz      EndIC           ;end if zero
096B FF C7                      158             inc     di              ;go to next character
096D FF C7                      159             inc     di              ;word/wide
096F EB E5                      160             jmp     NC              ;go to next character
                                161
                                162     EndIC:
0971 8B 3E 38 20                163             mov     di,WritePtr     ;get writePointer for insertion
0975 D1 E7                      164             shl     di              ;into word buffer
0977 89 95 3A 20                165             mov     KbdBuffer!di,dx ;save the word
097B D1 EF                      166             shr     di              ;undo shift
097D FF C7                      167             inc     di              ;inc write ptr tentatively
097F 83 FF 20                   168             cmp     di,##CBLength   ;wrap around
0982 75 03                      169             jne     IncWP
0984 BF 00 00                   170             mov     di,##0
                                171     IncWP:
0987 3B 3E 36 20                172             cmp     di,ReadPtr      ;Has the writepointer caught up
098B 74 04                      173             je      NewBit          ;yes, don't inc after all
098D 89 3E 38 20                174             mov     WritePtr, di    ;new writepointer
                                175     NewBit:
0991 D1 26 04 10                176             shlw    Mask            ;shift the bit left in the word,
0995 E2 9D                      177             loop    NextBit         ;and loop again
0997 E9 7C FF                   178             jmp     NewSetOfBits    ;otherwise get new
                                179
                                180     EndKbd:
099A A1 38 20                   181             mov     ax,WritePtr
099D 2B 06 36 20                182             sub     ax,ReadPtr      ;subtract read from write ptr.
09A1 7D 03                      183             jge     NoAdd           ;if it's positive, ok
09A3 05 20 00                   184             add     ax,#CBlength    ;otherwise add buffer length
                                185     NoAdd:
09A6 A2 19 20                   186             mov     KbdStatus+1,al  ;word of status(#chars in buf)
                                187
09A9 C3                         188             ret                     ;return to caller
                                189
                                190             .END
                                 32
                                 33             .Get    "Z-Cursor.bca"          ;move the cursor
                                  1     ;       Z-Cursor.bca
                                  2
                                  3     ;       Update the cursor coordinates and move the cursor
                                  4
                                  5     ;       Author: Bruce Horn
                                  6     ;       Last changed: April 20, 1979  12:24 PM
                                  7
                                  8
                                  9     DoCursor:
09AA 2E                           10            seg     cs
```

```
09AB A1 62 21                11          mov     ax,CursorLock   ;should I ignore this cursor?
09AE 3D 00 00                12          cmp     ax,#0
09B1 74 02                   13          je      Normal          ;no--it's ok
09B3 7C 1F                   14          jl      NCR             ;if less than zero, no cursor
                             15
                             16  Normal:
09B5 B8 00 00                17          mov     ax,#CurrentDS
09B8 8E D8                   18          mov     ds,ax
09BA 8E C0                   19          mov     es,ax
09BC A1 26 20                20          mov     ax,OldMX
09BF 3B 06 1A 20             21          cmp     ax,MouseX       ;has mouse x loc changed?
09C3 75 09                   22          jne     CR              ;yes, must do cursor
09C5 A1 28 20                23          mov     ax,OldMY
09C8 3B 06 1C 20             24          cmp     ax,MouseY       ;has mouse Y changed?
09CC 74 06                   25          je      NCR             ;no, don't do cursor
                             26  CR:
09CE E8 04 00                27          call    BltOld          ;blt out old cursor
09D1 E8 0A 00                28          call    BltNew          ;then blt in new
                             29  NCR:
09D4 C3                      30          ret                     ;return to caller
                             31
                             32  BltOld:
09D5 2E                      33          seg     cs
09D6 C5 36 22 21             34          lds     si,.CursorTable ;get address of table
09DA E8 A8 F8                35          call    BitBlt          ;blt the cursor out
09DD C3                      36          ret
                             37  BltNew:
09DE B8 00 00                38          mov     ax,#CurrentDS
09E1 8E D8                   39          mov     ds,ax
09E3 8E C0                   40          mov     es,ax
09E5 A1 1A 20                41          mov     ax,MouseX       ;BitBLT saves DS
09E8 A3 26 20                42          mov     OldMX,ax
09EB 2B 06 5E 21             43          sub     ax,CursorXOff
09EF A3 3E 21                44          mov     CursorX,ax
09F2 A1 1C 20                45          mov     ax,MouseY
09F5 A3 28 20                46          mov     OldMY,ax
09F8 2B 06 60 21             47          sub     ax,CursorYOff
09FC A3 40 21                48          mov     CursorY,ax
09FF C5 36 22 21             49          lds     si,.CursorTable ;Get cursor table
0A03 E8 7F F8                50          call    BitBlt          ;blt new cursor in at new pos
0A06 C3                      51          ret
                             52
                             53          .END
                             34
                             35          .Get    "Z-DRW.bca"             ;disk code
                              1  ;       Z-DRW.bca
                              2
                              3  ;       This program loads and stores core images on the floppy
                              4  ;       disk in a fixed format:  128 bytes/sector, 16 sectors/track.
                              5  ;       For read, it is given a starting address and an ending
                              6  ;       address to load--it assumes that the correct disk is in, and
                              7  ;       that it starts at track zero, sector zero.
                              8  ;       Author: Bruce Horn
                              9  ;       Last Edited: May 7, 1979  2:00 AM
                             10
                             11  ;Disk IO ports
0120                         12  Port =          0120    ;Address of controller
0120                         13  StatReg =       Port    ;Status Register in WD 1791
0120                         14  CmdReg =        Port    ;Command Register
0122                         15  TrackReg =      Port+2  ;Track Register
0124                         16  SectorReg =     Port+4  ;Sector Register
0126                         17  DataReg =       Port+6  ;Data Register
                             18
                             19  ;Disk commands
00FC                         20  RstCmd=         0FC     ;Restore
0077                         21  RdCmd=          077     ;Read
0057                         22  WrCmd=          057     ;Write
00E8                         23  SeCmd=          0E8     ;Seek (w/verify--no verify, 0EC)
                             24
                             25  ;Status bits--Type I and Type II commands
0080                         26  NotReady=       080     ;disk not ready
0040                         27  WrtProtect=     040     ;Write protect tab on
0001                         28  Busy=           001     ;command in progress
                             29
                             30  ;Status bits--type I commands only
0020                         31  HdLoad=         020     ;Head loaded
```

```
0010                    32    SeekErr=        010     ;Track not verified
0004                    33    Track00=        004     ;Track zero
0002                    34    Index=          002     ;index mark passed
                        35
                        36    ;Status bits--type II commands only
0010                    37    RNF=            010     ;Record Not Found
0008                    38    CRCErr=         008     ;CRC check invalid
0004                    39    LostData=       004     ;8086 couldn't keep up with 1791
0002                    40    DataRQ=         002     ;1971 wants data
                        41
                        42    ;Non error status bits--mask off on read status--type I
                        43    ;Track00 or Index or WrtProtect or HdLoad or Busy
0067                    44    NonErrI=        067
                        45
                        46    ;Non error status bits--mask off on read status--type II
                        47    ;DataRQ or Busy
0003                    48    NonErrII=       003
                        49
                        50    ;Status bits for recoverable errors--Type I
                        51    ;SeekErr
0010                    52    RecErrI=        010
                        53
                        54    ;Status bits for recoverable errors--Type II
                        55    ;RNF or CRCErr or LostData
001C                    56    RecErrII=       01C
                        57
                        58
                        59    LoadFloppy:
0A07 2E                 60            seg     cs
0A08 A1 87 0D           61            mov     ax,.LoadDiskCursor
0A0B E8 35 FC           62            call    ShowNewCursor
0A0E E8 20 FC           63            call    WaitBug
0A11 2E                 64            seg     cs
0A12 A1 65 0D           65            mov     ax,.NormalCursor
0A15 E8 2B FC           66            call    ShowNewCursor
0A18 E8 92 FC           67            call    HideCursor
0A1B E8 EE FB           68            call    IntsOff         ;mask off interrupts
0A1E E8 00 FC           69            call    DisplayOff
0A21 B8 00 00           70            mov     ax,#0
0A24 8E D8              71            mov     ds,ax           ;Zero the segments
0A26 8E C0              72            mov     es,ax
0A28 E8 1B 01           73            call    DiskStart
                        74    LoadDisk:
0A2B B8 00 00           75            mov     ax,#0
0A2E 8E D8              76            mov     ds,ax           ;Zero the segments
0A30 8E C0              77            mov     es,ax
0A32 A2 56 10           78            mov     Track,al        ;the Track register,
0A35 FE C0              79            inc     al
0A37 A2 57 10           80            mov     Sector,al       ;and set sector to 1
0A3A 8B 3E 58 10        81            mov     di,.Buffer      ;Write first sector to Buffer
0A3E E8 B8 00           82            call    ReadTS
0A41 8B 36 58 10        83            mov     si,.Buffer
0A45 8B 04              84            mov     ax,00!si        ;Byte address
0A47 A3 48 10           85            mov     NextDisk,ax     ;Continuation disk needed?
0A4A 8B 44 02           86            mov     ax,02!si
0A4D A3 4C 10           87            mov     BeginAdr,ax     ;Load from this address
0A50 A3 50 10           88            mov     OldAdr,ax
0A53 8B F8              89            mov     di,ax
0A55 8B 44 04           90            mov     ax,04!si
0A58 A3 4E 10           91            mov     BeginSeg,ax     ;in this segment
0A5B A3 52 10           92            mov     OldSeg,ax
0A5E 8E C0              93            mov     es,ax
0A60 8B 44 06           94            mov     ax,06!si
0A63 A3 54 10           95            mov     Blocks,ax       ;this many blocks
0A66 8B 44 08           96            mov     ax,08!si
0A69 A3 44 10           97            mov     JumpTo,ax       ;and when done, jump to
0A6C C7 06 4A 10 08 00  98            mov     ReTry,##8.      ;8 retries on dsk error
                        99    ReadNext:
0A72 FE 06 57 10       100            inc     Sector
0A76 FE 06 57 10       101            inc     Sector
0A7A E8 7C 00          102            call    ReadTS
0A7D E8 0B 01          103            call    DiskWait        ;wait for status ok
0A80 BA 20 01          104            mov     dx,#StatReg
0A83 EC                105            ind
0A84 34 FF             106            xor     al,#0FF
0A86 A8 1C             107            test    al,#RecErrII    ;recoverable error?
```

```
0A88 74 1B              108             jz      SectorOK        ;no, its ok
0A8A A1 50 10           109             mov     ax,OldAdr       ;restore old address
0A8D 8B F8              110             mov     di,ax
0A8F A1 52 10           111             mov     ax,OldSeg       ;and old segment
0A92 8E C0              112             mov     es,ax
0A94 FE 0E 57 10        113             dec     Sector          ;for later INC
0A98 FE 0E 57 10        114             dec     Sector
0A9C FF 0E 4A 10        115             decw    ReTry
0AA0 75 D0              116             jnz     ReadNext
0AA2 E9 38 01           117             jmp     DiskFail
                        118     SectorOK:
0AA5 8B C7              119             mov     ax,di
0AA7 8C C3              120             mov     bx,es
0AA9 E8 1B 01           121             call    NextAdr
0AAC 8E C3              122             mov     es,bx
0AAE 8B F8              123             mov     di,ax
0AB0 FF 0E 54 10        124             decw    Blocks
0AB4 75 24              125             jnz     RNS
0AB6 A1 48 10           126             mov     ax,NextDisk
0AB9 3D 01 00           127             cmp     ax,#1
0ABC EB 09              128             j       DR              ;was jne DR (changed for test)
0ABE E8 57 FB           129             call    IntsOn
0AC1 E8 65 FB           130             call    DisplayOn
0AC4 E9 40 FF           131             jmp     LoadFloppy      ;next disk flag on, otherwise
0AC7 C7 06 42 10 00 00  132     DR:     mov     DStatus,##0
0ACD E8 A1 00           133             call    DiskStop        ;deselect drive
0AD0 E8 45 FB           134             call    IntsOn
0AD3 E8 53 FB           135             call    DisplayOn
0AD6 E8 E7 FB           136             call    ShowCursor
0AD9 C3                 137             ret                     ;Return to caller
0ADA A0 57 10           138     RNS:    mov     al,Sector
0ADD 3C 0F              139             cmp     al,#15.
0ADF 7C 91              140             jl      ReadNext
0AE1 7F 07              141             jg      Q1
0AE3 C6 06 57 10 00     142             mov     Sector,#0
0AE8 EB 88              143             jmp     ReadNext
0AEA FE 06 56 10        144     Q1:     inc     Track
0AEE E8 27 00           145             call    Seek
0AF1 C6 06 57 10 FF     146             mov     Sector,#0FF     ;-1, with 2 incs, gives 1
0AF6 E9 79 FF           147             jmp     ReadNext
                        148
                        149
                        150     ReadTS:
0AF9 B3 77              151             mov     bl,#RdCmd
0AFB E8 AC 00           152             call    DiskSetup
0AFE B4 00              153             mov     ah,#0
0B00 EB 09              154             j       ReadLoop
                        155     ReadData:
0B02 BA 26 01           156             mov     dx,#DataReg
0B05 EC                 157             ind
0B06 34 FF              158             xor     al,#0FF         ;invert from WD bus
0B08 AA                 159             stob                    ;store into di/es
0B09 FE C4              160             inc     ah
                        161     ReadLoop:
0B0B BA 20 01           162             mov     dx,#StatReg
0B0E EC                 163             ind
0B0F A8 02              164             test    al,#DataRQ              ;Data request?
0B11 74 EF              165             jz      ReadData        ;yes, read more data
0B13 A8 01              166             test    al,#Busy        ;is it still busy?
0B15 74 F4              167             jz      ReadLoop        ;yes, wait again
0B17 C3                 168             ret
                        169
                        170     Seek:
0B18 2E                 171             seg     cs
0B19 C7 06 4A 10 08 00  172             mov     ReTry,##8.
0B1F 2E                 173             seg     cs
0B20 A0 56 10           174             mov     al,Track
0B23 34 FF              175             xor     al,#0FF         ;Invert for WD bus
0B25 BA 26 01           176             mov     dx,#DataReg
0B28 EE                 177             outd                    ;send it track number
0B29 B0 E8              178             mov     al,#SeCmd
0B2B BA 20 01           179             mov     dx,#CmdReg
0B2E EE                 180             outd                    ;Do seek
0B2F E8 59 00           181             call    DiskWait        ;wait for dsk to be ready
0B32 BA 20 01           182             mov     dx,#StatReg
0B35 EC                 183             ind
```

```
0B36 34 FF              184             xor     al,#0FF
0B38 A8 10              185             test    al,#RecErrI
0B3A 75 01              186             jnz     SeekError
0B3C C3                 187             ret
                        188     SeekError:
0B3D FF 0E 4A 10        189             decw    ReTry
0B41 75 D5              190             jnz     Seek
0B43 E9 97 00           191             jmp     DiskFail
                        192
                        193
                        194
                        195     DiskStart:
0B46 B8 04 00           196             mov     ax,#4
0B49 BB C7 FF           197             mov     bx,#DiskDrive
0B4C E8 07 FB           198             call    SetP100
0B4F E8 AA FA           199             call    WaitASecond     ;wait for the disk to spin up
0B52 B8 00 00           200             mov     ax,#0
0B55 BB FF EF           201             mov     bx,#DiskMR
0B58 E8 FB FA           202             call    SetP100
0B5B E8 9E FA           203             call    WaitASecond     ;wait a second on disk-mr
0B5E B8 01 00           204             mov     ax,#1
0B61 BB FF EF           205             mov     bx,#DiskMR
0B64 E8 EF FA           206             call    SetP100
0B67 E8 21 00           207             call    DiskWait        ;disk MR may restore disk
0B6A E8 8F FA           208             call    WaitASecond
0B6D E8 11 00           209             call    DiskRestore     ;but if it doesn't...
0B70 C3                 210             ret
                        211
                        212     DiskStop:
0B71 E8 17 00           213             call    DiskWait
0B74 E8 0A 00           214             call    DiskRestore
0B77 B8 00 00           215             mov     ax,#0
0B7A BB C7 FF           216             mov     bx,#DiskDrive
0B7D E8 D6 FA           217             call    SetP100         ;deselect drive
0B80 C3                 218             ret
                        219
                        220     DiskRestore:
0B81 B0 FC              221             mov     al,#RstCmd
0B83 BA 20 01           222             mov     dx,#CmdReg      ;send command
0B86 EE                 223             outd
0B87 E8 01 00           224             call    DiskWait        ;wait for it to be ready
0B8A C3                 225             ret
                        226
                        227     DiskWait:
0B8B BA 20 01           228             mov     dx,#StatReg     ;get status register contents
0B8E EC                 229             ind
0B8F A8 01              230             test    al,#Busy        ;is it still busy?
0B91 74 F8              231             jz      DiskWait        ;yes, try again,
0B93 C3                 232             ret                     ;otherwise return
                        233
                        234     DiskPwrOn:
0B94 B8 02 00           235             mov     ax,#2
0B97 BB FC FF           236             mov     bx,#DiskPower   ;Turn on first +5, then +12
0B9A E8 B9 FA           237             call    SetP100
0B9D E8 5C FA           238             call    WaitASecond
0BA0 B8 03 00           239     mov     ax,#3
0BA3 BB FC FF           240             mov     bx,#DiskPower
0BA6 E8 AD FA           241             call    SetP100
0BA9 C3                 242             ret
                        243
                        244     DiskSetup:
0BAA E8 DE FF           245             call    DiskWait        ;wait for disk to be ready
0BAD FC                 246             cld                     ;clear direction
0BAE 2E                 247             seg     cs
0BAF A0 57 10           248             mov     al,Sector       ;set sector
0BB2 34 FF              249             xor     al,#0FF         ;invert for WD bus
0BB4 BA 24 01           250             mov     dx,#SectorReg
0BB7 EE                 251             outd
0BB8 8A C3              252             mov     al,bl           ;get command number
0BBA BA 20 01           253             mov     dx,#CmdReg      ;send command
0BBD EE                 254             outd
                        255     WaitBusy:
0BBE BA 20 01           256             mov     dx,#StatReg     ;get status register contents
0BC1 EC                 257             ind
0BC2 A8 01              258             test    al,#Busy        ;is it busy yet?
0BC4 74 F8              259             jz      WaitBusy        ;command not yet executing
```

```
0BC6 C3                 260             ret
                        261
                        262     ;ax=byte address, bx=segment address
                        263     NextAdr:
0BC7 8B D0              264             mov     dx,ax           ;get byte address
0BC9 B9 04 00           265             mov     cx,#4           ;and shift right by four for
0BCC D3 EA              266             shrv    dx              ;segment address to be added to
0BCE 03 DA              267             add     bx,dx           ;old segment register
0BD0 25 0F 00           268             and     ax,#0F          ;save low 4 bits of byte address
0BD3 2E                 269             seg     cs
0BD4 A3 50 10           270             mov     OldAdr,ax
0BD7 2E                 271             seg     cs
0BD8 89 1E 52 10        272             mov     OldSeg,bx
0BDC C3                 273             ret
                        274
                        275     DiskFail:
0BDD B8 0C 00           276             mov     ax,#0C
0BE0 BB F0 FF           277             mov     bx,#LedsOn
0BE3 E8 7A FA           278             call    SetP20
0BE6 FB                 279             sti                     ;set interrupts
0BE7 BA 20 01           280             mov     dx,#StatReg
0BEA EC                 281             ind                     ;get status register
0BEB 34 FF              282             xor     al,#0FF         ;invert for WD bus
0BED 2E                 283             seg     cs
0BEE A3 42 10           284             mov     DStatus,ax      ;save it
0BF1 E8 24 FA           285             call    IntsOn
0BF4 E8 32 FA           286             call    DisplayOn
0BF7 2E                 287             seg     cs
0BF8 A1 A9 0D           288             mov     ax,.ErrorCursor
0BFB E8 45 FA           289             call    ShowNewCursor
0BFE EB FE              290     DF:     j       DF              ;halt machine
                        291
                        292                     .End
                         36
                         37             .Get    "Z-RI.bca"              ;rom initialization
                          1     ;       Z-RI.bca
                          2     ;       This file contains all the necessary tables for
                          3     ;       normal execution in rom.  Page 1 locations (at 2000H)
                          4     ;       will be initialized by a rep/movw sequence.
                          5
                          6     ;       Author:  Bruce Horn
                          7     ;       Last edited: May 7, 1979  12:34 AM
                          8
21A0                      9     CursorToBe=     21A0            ;location of cursor bitmap
0017                     10     CBTblLength=    23.             ;in words, including offsets/lock
0010                     11     CBMapLength=    16.             ;in words
                         12
                         13     RomInit:
                         14
0C00 B8 00 00            15             mov     ax,#CurrentDS
0C03 8E D8               16             mov     ds,ax
0C05 8E C0               17             mov     es,ax
0C07 FC                  18             cld                     ;incrementing
0C08 B9 17 00            19             mov     cx,#CBTblLength ;how many words to move
0C0B 8B 36 13 0D         20             mov     si,.CBTbl
0C0F BF 36 21            21             mov     di,#CursorTable ;address of table to be
0C12 89 3E 22 21         22             mov     .CursorTable,di ;save address in loc
0C16 A3 24 21            23             mov     .CursorTable+2,ax       ;and its segment
0C19 F2                  24             rep
0C1A A5                  25             movw                    ;and blt table up
                         26
0C1B B9 10 00            27             mov     cx,#CBMapLength ;length of cursor bitmap
0C1E 8B 36 43 0D         28             mov     si,.Crsr        ;address of cursor
0C22 BF A0 21            29             mov     di,#CursorToBe  ;and its destination
0C25 F2                  30             rep
0C26 A5                  31             movw                    ;blt table up
                         32
                         33     ;Zero registers, keyboard data blocks...
                         34
0C27 B8 00 00            35             mov     ax,#0           ;to zero registers
                         36
0C2A B9 06 00            37             mov     cx,#6.
0C2D BF 2A 20            38             mov     di,#KbdData     ;clear KbdData block...
0C30 F2                  39             rep
0C31 AB                  40             stow
0C32 B9 06 00            41             mov     cx,#6.
```

```
0C35 BF 14 10                  42              mov     di,#OldKbdData  ;and OldKbdData block...
0C38 F2                        43              rep
0C39 AB                        44              stow
                               45
0C3A C7 06 0C 10 03 00         46              mov     Port100,##3     ;initial port 100 bits
0C40 C7 06 0E 10 00 00         47              mov     Port60,##0      ;and port60 bits
0C46 C7 06 10 10 88 00         48              mov     Port20,##88     ;and port 20 bits
0C4C C7 06 12 10 0C 00         49              mov     KbdOffset,##12. ;KbdInt won't hit before OddInt
0C52 A3 26 20                  50              mov     OldMX,ax        ;last MX
0C55 A3 28 20                  51              mov     OldMY,ax        ;last MY
0C58 A3 1A 20                  52              mov     MouseX,ax       ;MouseX
0C5B A3 1C 20                  53              mov     MouseY,ax       ;MouseY
0C5E A3 36 20                  54              mov     ReadPtr,ax      ;ReadPtr
0C61 A3 38 20                  55              mov     WritePtr,ax     ;WritePtr
0C64 A3 18 20                  56              mov     KbdStatus,ax    ;Keyboard Status
0C67 A3 14 20                  57              mov     ICount,ax       ;interrupt ctr.
0C6A A3 0C 21                  58              mov     NIChars,ax      ;interrupt chars
0C6D A3 1E 20                  59              mov     RMX,ax          ;lower limit of mouse x,y
0C70 A3 20 20                  60              mov     RMY,ax          ;is zero
0C73 C7 06 00 20 85 02         61              mov     .BitBlt,##BitBlt        ;address of BitBlt
0C79 C7 06 02 20 56 06         62              mov     .SetP100,##SetP100      ;address of SetP100
0C7F C7 06 04 20 60 06         63              mov     .SetP20,##SetP20
0C85 C7 06 06 20 99 06         64              mov     .InitDC,##InitDC
0C8B C7 06 08 20 AD 06         65              mov     .HideCursor,##HideCursor
0C91 C7 06 0A 20 C0 06         66              mov     .Showcursor,##ShowCursor
0C97 C7 06 0C 20 00 0C         67              mov     .RomInit,##RomInit
0C9D C7 06 30 10 43 06         68              mov     .ShowNewCursor,##ShowNewCursor
0CA3 C7 06 32 10 29 06         69              mov     .DisplayOn,##DisplayOn
0CA9 C7 06 34 10 21 06         70              mov     .DisplayOff,##DisplayOff
0CAF C7 06 36 10 31 06         71              mov     .WaitBug,##WaitBug
0CB5 C7 06 38 10 3A 06         72              mov     .WaitNoBug,##WaitNoBug
0CBB C7 06 3A 10 18 06         73              mov     .IntsOn,##IntsOn
0CC1 C7 06 3C 10 0C 06         74              mov     .IntsOff,##IntsOff
                               75
                               76      ;Clear screen
0CC7 C5 36 22 21               77              lds     si,.CursorTable ;get table for bitblt
0CCB 8B 44 14                  78              mov     ax,ClipW!si     ;upper limit of mouse x,y
0CCE A3 22 20                  79              mov     RMW,ax          ;is clipping corner
0CD1 8B 44 16                  80              mov     ax,ClipH!si
0CD4 A3 24 20                  81              mov     RMH,ax
0CD7 B8 80 02                  82              mov     ax,#640.        ;full width
0CDA 89 44 0C                  83              mov     DestW!si,ax     ;for bitblt table
0CDD B8 E0 01                  84              mov     ax,#480.        ;and full height
0CE0 89 44 0E                  85              mov     DestH!si,ax     ;before screen clear
0CE3 B8 0C 00                  86              mov     ax,#StcBB
0CE6 89 44 24                  87              mov     Function!si,ax
0CE9 B8 00 00                  88              mov     ax,#White
0CEC 89 44 26                  89              mov     GrayBits!si,ax
0CEF E8 93 F5                  90              call    BitBlt          ;clear screen white
0CF2 2E                        91              seg     cs
0CF3 C5 36 22 21               92              lds     si,.CursorTable
0CF7 B8 10 00                  93              mov     ax,#16.         ;cursor width
0CFA 89 44 0C                  94              mov     DestW!si,ax
0CFD 89 44 0E                  95              mov     DestH!si,ax
0D00 B8 02 00                  96              mov     ax,#XorBB
0D03 89 44 24                  97              mov     Function!si,ax  ;make sure function is xor
0D06 B8 FF FF                  98              mov     ax,#Black       ;with all ones
0D09 89 44 26                  99              mov     GrayBits!si,ax
0D0C E8 EF F9                 100              call    IOInit          ;turn on CRT, keyboard
0D0F E8 AE F9                 101              call    ShowCursor      ;now put up first cursor
0D12 C3                       102              ret                     ;and return to caller
                              103
0D13 15 0D                    104      .CBTbl: .adr    CBTbl
                              105
                              106      CBTbl:
0D15 00 40                    107              .adr    4000            ;byte - destination bitmap
0D17 00 00                    108              .adr    0               ;seg
0D19 28 00                    109              .adr    40.             ;raster
0D1B 80 25                    110              .adr    240.*40.        ;field size
0D1D 00 00                    111              .adr    0.              ;x - destination rectangle
0D1F 00 00                    112              .adr    0.              ;y
0D21 10 00                    113              .adr    16.             ;width
0D23 10 00                    114              .adr    16.             ;height
0D25 00 00                    115              .adr    0.              ;x - clipping rectangle
0D27 00 00                    116              .adr    0.              ;y
0D29 80 02                    117              .adr    640.            ;width
```

```
0D2B E0 01              118             .adr    480.            ;height
0D2D A0 21              119             .adr    CursorToBe      ;byte - source bitmap
0D2F 00 00              120             .adr    CurrentDS       ;seg
0D31 01 00              121             .adr    1.              ;raster
0D33 08 00              122             .adr    8.              ;field size
0D35 00 00              123             .adr    0.              ;x - source origin
0D37 00 00              124             .adr    0.              ;y
0D39 02 00              125             .adr    02              ; (xor)
0D3B FF FF              126             .adr    0FFFF           ;black
                        127
0D3D 00 00              128             .adr    0               ;offset x
0D3F 00 00              129             .adr    0               ;offset y
                        130
0D41 FF FF              131             .adr    0FFFF           ;cursor lock (<0 means locked)
                        132
0D43 45 0D              133     .Crsr:  .adr    Cursor
                        134
                        135     Cursor:
0D45 00 FF              136             .adr    0FF00           ;line 0
0D47 00 FE              137             .adr    0FE00           ;line 2
0D49 00 FC              138             .adr    0FC00           ;line 4
0D4B 00 E7              139             .adr    0E700           ;line 6
0D4D FF 01              140             .adr    001FF           ;line 8
0D4F 00 00              141             .adr    00000           ;line 10
0D51 86 C8              142             .adr    0C886           ;line 12
0D53 80 98              143             .adr    09880           ;line 14
0D55 00 FF              144             .adr    0FF00           ;line 1
0D57 00 FC              145             .adr    0FC00           ;line 3
0D59 00 FE              146             .adr    0FE00           ;line 5
0D5B 80 C3              147             .adr    0C380           ;line 7
0D5D FF 00              148             .adr    000FF           ;line 9
0D5F E6 8B              149             .adr    08BE6           ;line 11
0D61 86 A8              150             .adr    0A886           ;line 13
0D63 86 88              151             .adr    08886           ;line 15 of cursor
                        152
                        153
                        154
                        155             .end
                         38
                         39             .Get    "Z-CursorBitmaps.bca"   ;Cursors
                          1     ;       Z-CursorBitmaps.bca
                          2
                          3     ;       Contains various cursor bitmaps for
                          4     ;       sending low-level messages to user
                          5     ;       Author: Bruce Horn
                          6     ;       Last changed: May 7, 1979  1:56 AM
                          7
                          8
                          9     .NormalCursor:
0D65 67 0D                10             .adr    NormalCursor
                         11
                         12     NormalCursor:
0D67 00 FF               13             .adr    0FF00           ;line 0
0D69 00 FE               14             .adr    0FE00           ;line 2
0D6B 00 FC               15             .adr    0FC00           ;line 4
0D6D 00 E7               16             .adr    0E700           ;line 6
0D6F FF 01               17             .adr    001FF           ;line 8
0D71 00 00               18             .adr    00000           ;line 10
0D73 86 C8               19             .adr    0C886           ;line 12
0D75 80 98               20             .adr    09880           ;line 14
0D77 00 FF               21             .adr    0FF00           ;line 1
0D79 00 FC               22             .adr    0FC00           ;line 3
0D7B 00 FE               23             .adr    0FE00           ;line 5
0D7D 80 C3               24             .adr    0C380           ;line 7
0D7F FF 00               25             .adr    000FF           ;line 9
0D81 E6 8B               26             .adr    08BE6           ;line 11
0D83 86 A8               27             .adr    0A886           ;line 13
0D85 86 88               28             .adr    08886           ;line 15 of cursor
                         29
                         30     .LoadDiskCursor:
0D87 89 0D               31             .adr    LoadDiskCursor
                         32
                         33     LoadDiskCursor:
0D89 00 00               34             .adr    00000           ;line 0
0D8B AA 8A               35             .adr    08AAA           ;line 2
0D8D AA 8A               36             .adr    08AAA           ;line 4
```

```
0D8F 00 00              37              .adr    00000           ;line 6
0D91 EA CE              38              .adr    0CEEA           ;line 8
0D93 4C A4              39              .adr    0A44C           ;line 10
0D95 EA EE              40              .adr    0EEEA           ;line 12
0D97 B0 0D              41              .adr    00DB0           ;line 14
0D99 EC 8E              42              .adr    08EEC           ;line 1
0D9B EA 8A              43              .adr    08AEA           ;line 3
0D9D AE EE              44              .adr    0EEAE           ;line 5
0D9F 00 00              45              .adr    00000           ;line 7
0DA1 8C A4              46              .adr    0A48C           ;line 9
0DA3 2A A4              47              .adr    0A42A           ;line 11
0DA5 00 00              48              .adr    00000           ;line 13
0DA7 B0 0D              49              .adr    00DB0           ;line 15 of cursor
                        50
                        51      .ErrorCursor:
0DA9 AB 0D              52              .adr    ErrorCursor
                        53
                        54      ErrorCursor:
0DAB FF 00              55              .adr    000FF           ;line 0
0DAD FE 00              56              .adr    000FE           ;line 2
0DAF FC 00              57              .adr    000FC           ;line 4
0DB1 E7 00              58              .adr    000E7           ;line 6
0DB3 01 FF              59              .adr    0FF01           ;line 8
0DB5 00 00              60              .adr    00000           ;line 10
0DB7 C8 86              61              .adr    086C8           ;line 12
0DB9 98 80              62              .adr    08098           ;line 14
0DBB FF 00              63              .adr    000FF           ;line 1
0DBD FC 00              64              .adr    000FC           ;line 3
0DBF FE 00              65              .adr    000FE           ;line 5
0DC1 C3 80              66              .adr    080C3           ;line 7
0DC3 00 FF              67              .adr    0FF00           ;line 9
0DC5 8B E6              68              .adr    0E68B           ;line 11
0DC7 A8 86              69              .adr    086A8           ;line 13
0DC9 88 86              70              .adr    08688           ;line 15 of cursor
                        71
                        72
                        73
                        74
                        75
                        76
                        77
                        78
                        79              .END
                        40
                        41              .END
```