DBASE III Debugging procedure

JP 9/16/84

Make sure your default drive contains all necessary file(s).

Track 39. Side Ø.

Official Disk	My Disk
Sector 1 = All F6	All F6
Sector 2 = All F6	Random
Sector 3 = error	All F6 - ØØ in 1st byte
Sector 4 = All F6 - ØØ in 1st byte	All F6
Sector 5 = All F6	All F6

```
1. G Ø16D
2. A 1A2E        JMP   1A3C
3. G Ø2EF
4. A 141C        JMP   1428
5. G Ø2F8   (or G 74D in patched DBASE)
6. A Ø13Ø        INC   BYTE PTR [Ø14D]
                 CMP   BYTE PTR [Ø14D],8
                 JZ    Ø142
                 CALL  [BX+Ø9C6]
     Ø13F        JMP   181C
                 PUSHF
                 PUSH  CS
     Ø144        PUSH  [Ø14E]
                       5790:8E6
     Ø148        JMP   G9Ø:Ø8E6   ; examine last 4 bytes of 1st line of DØ:Ø
     Ø14D        DB    Ø          to determine correct address
                 DW    Ø13C       (E6 Ø8 DB 6C)
                                  on other system
7. A 1818        JMP   Ø13Ø       (E6 Ø8 9Ø 57)

8. G
9. G 15A5        ; first drive access occurs, position determined via "READ ID" command
                   for drive Ø, head Ø in MF mode
10. G 15AA       ; 2nd drive access occurs, "READ DATA" issued with multi-track bit
                   and SK bit (skip deleted data address mark)
11. R F          CY ✓      }
12. ~~R CX~~     ~~2203~~  }  these steps taken to mimic actual official operation
13. ~~R AX~~     ~~1ØØØ~~ ✓ }
14. G 15E1       ✓   18. R AX 1000 ✓           22. G 1673 (to post a few more [illegible] and 1 more access)
15. R F          NC ✓ 19. G 1632 (this finishes REPZ SCA)   23. ~~R F NC~~ (this is ok, though)
16. G 16ØF       ✓   20. R IP 1647 (skips JB & JA and points us to [illegible]) (trace a few) 4 times!
17. R F          CY ✓ 21. G 165C (takes us to end of a LOOP and to a CMP [illegible])   24. Breakpoint routine restore →
                     21A. A 1662 NOP NOP (wipes out the JNZ after the CMP above)
```

Breakpoint Routine Restore

```
A 015D:  PUSHF
         PUSH  DS
         PUSH  AX
         XOR   AX,AX
         MOV   DS,AX
         MOV   word ptr [4],1742
                            5AAA
         MOV   word ptr [6],~~5AAA~~   ; varies
         MOV   word ptr [C],FF54
                            5AAF
         MOV   word ptr [E],~~5AAF~~
         POP   AX
         POP   DS
         POPF
015D:172 JMP   181C
```

Then R IP 15Ø (only when the IP is already 13F)
Then G

DBASE III trace

JP 9/11/84
1st successful run: 9/21/84

```
64AA 0130  MOV  CS:[0170],AX      0 -> 0170   1st successful patch: who cares?
           MOV  CS:[0172],DS      649A -> 0172
           MOV  CS:[0174],ES      649A -> 0174
           CLI
           PUSH ES                649A on stack
           XOR  AX,AX             ; AX=0
           PUSH AX                0 on stack
           MOV  AX,CS             ; AX= 64AA
           MOV  DS,AX
           MOV  ES,AX             ; ES & DS also 64AA segment (code)
           STI
           MOV  BX,[0176]         ; it contains F79A
014E       MOV  SI,180            ;
           MOV  CX,[0178]         ; it contains 5000
           MOV  DI,SI             ; DI = 180
           SHR  CX,1              ; CX = 2800
           DEC  CX                ;    = 27FF
           MOV  DX,[017A]         ; it contains 657B
           CLD
           LODSW                  ;
0160       LODSW                  ; load starting from 182 (this segment)
           SUB  DX,AX             ; DX = DX - AX
           XCHG DX,AX             ; DX = word, AX = DX-AX
           XOR  AX,BX             ; exclusive-OR with F79A
           STOSW                  ; store starting at 180 (this segment)
           LOOP 0160              ; code from 64AA:0180 to 64AA: 517F recreated
           XCHG DX,AX
           XOR  AX,BX
           STOSW                  ; re-create last word
<BP> 016D  JMP  02D0
```

```
A013Ø   PUSHF
        PUSH  ES
        PUSH  AX
        XOR   AX,AX
        MOV   ES,AX
        ES:
        MOV   AX,[000E]
        PUSH  AX
        ES:
        MOV   AX,[000C]
        PUSH  AX
        ES:
        MOV   WORD PTR [000E],619Ø
        ES:
        MOV   WORD PTR [000C],Ø8E6
Ø14F    NOP
        POP   AX
        ES:
        MOV   [000C],AX
        POP   AX
        ES:
        MOV   [000E],AX
        POP   AX
        POP   ES
        POPF
Ø15D    <overwritten instruction(s)>
        JMP   <continuation point>
```

```
G016D
A1A2E  14 NOPs or JMP 1A3C
G02EF
A141C  12 NOPs or JMP 1428
G02F8
T
AØEE4  MOV AX,SS | MOV ES,AX | JMP ØF15
AØF18  MOV CX,Ø5 | NOP
GØF27
```

```
AØ13Ø: PUSHF
       PUSH  CS
       PUSH  BP
       PUSH  BP
       PUSH  BX
       MOV   BP,SP
       MOV   BX,[BP+ØA]
       MOV   [BP+Ø4],BX
       POP   BX
       POP   BP
       JMP   619Ø:Ø8E6
```

Ø144:

and CALL 13Ø should act like a Breakpoint interrupt

```
0200    CLD
        MOV     AH,30
        INT     21              ;get version?
        XCHG    AH,AL
        MOV     [0182],AX       ;save version?
        *CALL   0FA8
64AA 0FA8   PUSH    ES
        PUSH    DS
        PUSH    DS
        POP     ES              ;ES same as DS (64AA)
        XOR     AX,AX           ;AX = 0
        MOV     DS,AX           ;Absolute 0-page addressing via DS
        MOV     DI,0205             remember: DS is different now!
        MOV     BX,0
        CLD
        MOV     BP,CS:[01DF]    ;BP = 0008
        SHL     BP,1            ;BP = 16. (byte count?)
        MOV     CL,02
0FC0    MOV     DL,CS:[BX+01E1] ;DL = 27 initially
        XOR     DH,DH           ;DX = 27
        MOV     SI,DX           ;
        SHL     SI,CL           ;SI = 9C
        LODSW                   ;copy the 8 interrupt types specified:
        STOSW                       27 21 20 08 02 04 03 01
        LODSW                   to area at 0205 in this segment (64AA)
        STOSW
        ADD     BX,02
        CMP     BX,BP
        JLE     0FC0
        POP     DS
        POP     ES
        MOV     AX,19F6
        *RET
```

```
64AA  02DD   *CALL  AX              ; CALL 19F6
64AA  19F6    MOV   DI,140A
              MOV   CX,140D
              MOV   AH,[09F6]      ; AH = 57
             *CALL  1A25
64AA  1A25    MOV   SI,DI          ; SI = 140A
              SUB   CX,DI          ; CX = CD_H  (bytes to transfer)
              PUSH  CS
              XOR   BX,BX
              MOV   ES,BX          ; absolute addressing with ES!
(14 nops)     MOV   word ptr ES:[0004],1742   ; steering up Single-step vector
1A36          MOV   word ptr ES:[000C],09DB   ; steering up Breakpoint vector
1A3C          POP   ES
              JMP   1A40
1A40          LODSB                ;
              XOR   AL,AH          ; code from 140A to 140D recreated
              STOSB                ; according to mask (57 in this case)
              LOOP  1A40
1A46         *RET
64AA  1A03    . . .
1A24         *RET

64AA  02DF    MOV   Byte ptr CS:[02E2],02
              CMP   AX,0000        ; it is 0, and BX is 0E5A
              JZ    02ED           ;
              . . .
02ED          CALL  BX             ;
64AA  0E5A    (Convert some more code with 1A25 calls)
              MOV   BX,0EA8
              MOV   [0616],BX
              MOV   AX,140A
              RET
```

```
(BP) 64AA  02EF   *CALL  AX          ; which is 140A
           140A   MOV   DI,0200     ;
                  MOV   SI,DI
                  MOV   CX,02EF
                  SUB   CX,DI
           1414   LODSB             ; converting more code
                  XOR   AL,CL
                  STOSB
                  LOOP  1414
                  MOV   ES,CX
(76 nops)  141C   MOV   Byte ptr ES:[0004],CC   ; screwing up single-step again
           1422   MOV   Byte ptr ES:[000C],AA   ; screwing up Breakpoint again
           1428   PUSH  CS
                  CLD
                  MOV   AX,F000
                  MOV   ES,AX
                  MOV   DI,E00E
                  LEA   SI,[0958]   ; SI = 0958
                  MOV   CX,0003
                  REPZ
                  CMPSB             ; compare bytes at 64AA:0958 against F000:E00E
                  JNZ   1467          Should match as 'IBM'
                  MOV   SI,FFFE
                  CMP   Byte ptr ES:[SI],FF   ; should match FF
                  JNZ   144D                  ; oops
                  MOV   Byte ptr [022C],01    ; should go to 1 (from 2 earlier)
                  JZ    14AA
           144D   ...
           14AA   (convert more code)
                  POP   ES
                  MOV   SI,0230
                  *RET
```

```
64AA 02F1  CALL  DX
64AA 0F88  MOV   DI,0EA8
           MOV   CX,0F87
           MOV   AH,[0946]
           XOR   Byte ptr CS:[SI],CC   ; byte at 0230
           CALL  1A25
           MOV   DI,1742
           MOV   CX,1806
           CALL  1A25
           MOV   BX,0616
     0FA5  RET
64AA 02F3  CMP   BX,0          ; BX is 0616, 0EA8 is 0616
           JZ    02FA
(BP) 02F8  JMP   [BX]          ;

64AA 0EA8  CLI
           MOV   DI,0400
           MOV   CS:[0F70],SS
           MOV   CS:[0F72],SP
           XOR   AX,AX
           MOV   ES,AX         ; Absolute addr. via ES
           MOV   AX,ES:[0008]
           MOV   CS:[0F83],AX
           MOV   AX,ES:[000A]  ; NMI vector copied?
           MOV   CS:[0F85],AX
           PUSH  CS
           POP   ES            ; ES restored (64AA)
           MOV   AX,F800
           MOV   SS,AX
           MOV   SP,8014       ; Stack at absolute 00014H? (dangerous)
           MOV   AX,17A4
           MOV   DX,2823
```

Normal Breakpoint vector (for this load only):
Segment (hi word) = 619Ø
Offset (lo word) = ØBE6

Trap vector:
619Ø
ØBED


```
64AA  ØEDA   PUSH  CS              ; stack at 00012H? OVERFLOW segment is same na
             MOV   BX,ØB89
             ADD   BX,CS:[ØF14]
             PUSH  AX              ; stack at 00010H? OVERFLOW Offset is 17AC
ØEE4
MOV AX,SS    STC
MOV ES,AX    SBB   AX,BX
JMP ØF15     PUSH  BX              ; you did it now!
             MOV   CL,CS:[ØF76]    ; CL = Ø4
             SAR   AX,CL
             PUSH  AX
             POP   AX              ; stack at 0000EH?
             MOV   CX,CS:[ØF77]    ; CX = ØØØC
             MOV   SI,ØØ4Ø
             SHR   CX,1
             INC   CX              ; CX = 7
ØEFC         LODSW
             PUSH  AX
             LOOP  ØEFC
             MOV   CX,ØØ55         ; stack at 00000H?
ØF03         ADD   SP,Ø8           ; stack at 00008H?
             PUSH  CX              ; stomp on single-step again?
             PUSH  DX
             ADD   SP,ØC           ; stack at ØØØ1ØH?
             PUSH  SS
             POP   ES              ; stomp on breakpoint again? AND ES = F8ØØ
             ADD   SP,78           ; stack at 00084H?
             SUB   SP,Ø214         ; pointless, since we restore SS/SP below
ØF15         MOV   AX,CS:[ØF7Ø]
ØF18
MOV CX,ØØØ5  AND   CX,ØØØ7         ; CX = ØØØ5, ES = F8ØØ
NOP          STC
             MOV   SS,AX
             MOV   BX,CS:[ØF72]
```

```
64AA  0F24   MOV   SP,BX
             STI
<BP>  0F27   ROL   DI,CL          ; DI = 0400 before, 8000 after
             XOR   DI,0010                 8010 after
             SUB   Word ptr ES:[DI],01B2   ; Overflow vector = 17A4, now = 15F2
             PUSH  ES
             XOR   AX,AX
             MOV   ES,AX
             MOV   AX,CS:[0F83]
             MOV   ES:[0008],AX
             MOV   AX,CS:[0F85]
             MOV   ES:[000A],AX
             POP   ES             ; NMI restured, ES back to F800
             PUSH  CS
             POP   DS             ; DS to here (64AA)
             CLC
      0F4B   XOR   BX,BX
             MOV   CX,0018
             MOV   SI,0F79        ; look 0F79 - 0FA8
Two NOPs 0F53   SUB   Word ptr ES:[8004],10E1   ; BYE-BYE single-step
      0F5A   LODSW
      0F5B   SUB   Word ptr ES:[DI],00C8   ; Overflow vector = 15F2, then 157A,
AISD 614F ->0F60   ADD   BX,AX                then 1462, then 139A, then 12D2
             INTO
             LOOP  0F5A
<BP> 64AA    0F65   XOR   BX,1000          see Page 8
             JNS   0F4B
             CMP   AX,[SI]
```

```
64AA 12D2  ADD  SP,+0C6      ; SP = 012C
           STI
           MOV  AX,CS
           MOV  ES,AX
           MOV  DS,AX
           MOV  DI,140A
           MOV  CX,0020
           REPZ
           STOSW
           CLI
           PUSH CS:[0996]    ; = 0180 (sign flag, trap flag)
           PUSH CS           ; New segment (actually same) 64AA
           PUSH CS:[098E]    ; new PC = 18DE
           STI
           IRET              ; Trap vector should point to 64AA:1742
64AA 1742  MOV  CS:[09F2],BX ; BX = 8817 maybe
           MOV  BX,CS:[099E]
           MOV  CS:[09A0],BX
           SHL  BX,1
           SHR  BX,1
           SHL  BX,1
           SHR  BX,1
           MOV  CS:[BX+140A],DS
           MOV  CS:[BX+140C],ES
           MOV  CS:[BX+140E],AX
           MOV  AX,CS:[09F2]
           MOV  CS:[BX+1410],AX
           MOV  CS:[BX+1412],CX
           MOV  CS:[BX+1414],DX
           MOV  CS:[BX+1416],SI
           MOV  CS:[BX+1418],DI
```

6AAA: 15F2

6AAA:0F79 = 0003
0F7B 0196
0F7D 75F3
0F7F 108B + 778C = 8817

```
64AA  1785  SHR   BX,1
            SHR   BX,1
            SHR   BX,1
            POP   CS:[BX+Ø98E]
            POP   CS:[Ø9F2]
            POP   CS:[BX+Ø996]
            ADD   BX,+Ø2
            CMP   BX,+08
            JB    17A4
            XOR   BX,BX
17A4        SHR   BX,1
            MOV   CS:[Ø99E],BX
            STI
            PUSH  CS
            POP   DS
            INC   word ptr [Ø9AE]
            POP   DS
            PUSH  DS
            XOR   AX,AX
            MOV   ES,AX
            MOV   DI,Ø
            MOV   SI,BP
            LODSW
            STOSW
            MOV   BX,CS
            MOV   AX,BX
            STOSW
            ADD   DI,+Ø4
            MOV   CX,3
            TEST  Byte ptr CS:[Ø22C],Ø8
            JZ    17D6
```

```
64AA 17D2   ADD   DI,+04
            DEC   CX
     17D6   LODSB
            STOSB
            LODSB
            STOSB
            MOV   AX,CS:[09AE]
            AND   AX,7
            ADD   BX,AX
            MOV   ES:[DI],BX
            ADD   BX,CX
            ADD   DI,+02
            LOOP  17D6
            PUSH  CS
            POP   DS
     17EF   *CALL  187D
64AA 187D   MOV   BX,CS:[09A0]
            SHL   BX,1
            MOV   DI,[BX+09BE]
            SUB   DI,+07
            MOV   SI,DI
            MOV   AX,CS
            MOV   ES,AX
            MOV   DS,AX
            MOV   CX,05
     1896   LODSB
            XOR   AL,BL
            MOV   BL,AL
            ADD   BX,[09AE]
            STOSB
            LOOP  1896
            *RET
```

```
64AA 17EF  *CALL  18A3
64AA 18A3  PUSH   CS
           POP    DS
           MOV    BX,[099E]
           MOV    DL,[BX+09AA]
           SHL    BX,1
           MOV    DI,[BX+09AC]
           MOV    SI,DI
           MOV    AX,DI
           PUSH   CS
           POP    ES
           SUB    AX,[BX+099E]
           XCHG   AH,AL
           MOV    AL,07
           SUB    AL,AH
           XOR    AH,AH
           MOV    CX,AX
      18C7 LODSB
           XOR    AL,DL
           STOSB
           LOOP   18C7
           MOV    [BX+09AC],DI
           XOR    AX,AX
           MOV    DS,AX
          *RET
64AA 17F5  CMP    Word ptr [4],1742
           JZ     17FF
           CLI
           HLT
      17FF PUSH   CS
           POP    DS
```

```
64AA  18D1  MOV   AX, [09AE]
            MOV   CX, 16
            MOV   BX, FFFF
      18DA  INC   BX
            CMP   AL, [BX+09B0]
            LOOPNZ 18DA
            JNZ   182F
            PUSH  AX
            PUSH  BX
            PUSH  CX
            SHL   BX, 1
      1818  CALL  [BX+09C6]      ; CALL 130 point
      181C  CMP   AX, 0           BX = 18, call = 1577
            JZ    1825
            CALL  AX
            JMP   181C
      1825  POP   CX
CALL 130    POP   BX
            POP   AX
            PUSH  CS
            POP   DS
      182A  MOV   Byte ptr [BX+09B0], 0
      182F  JCXZ  1833
            JMP   18DA
      1833  XOR   AX, AX
            MOV   ES, AX
            MOV   word ptr ES:[4], 1742
            MOV   BX, [099E]
            SHL   BX, 1
            PUSH  CS:[BX+0996]
            PUSH  CS
            PUSH  CS:[BX+09BE]
```

Breakpoint Routine Restore:

```
0150:  PUSHF
       PUSH   DS
       PUSH   AX
       XOR    AX,AX
       MOV    DS,AX
       MOV    Word ptr[4],1742
       MOV    Word ptr[6],64AA
       MOV    Word ptr[C],FF54
       MOV    Word ptr[E],64AF
       POP    AX
       POP    DS
       POPF
       RET    or JMP 181C
```

these will vary - check and correct before:
rip 144 and 9
also, correct the return address on stack
and restore the instructions overwritten

```
64AA:184F  MOV   CL,03
           SHL   BX,CL
1853       MOV   ES,[BX+140C]
1857       MOV   AX,[BX+140E]
185B       MOV   CX,[BX+1410]
185F       MOV   [09F2],CX
1863       MOV   CX,[BX+1412]
1867       MOV   DX,[BX+1414]
186B       MOV   SI,[BX+1416]   ]  JMP  9000:0000
186F       MOV   DI,[BX+1418]   ]
1873       MOV   DS,[BX+140A]
           MOV   BX,CS:[09F2]
187C       IRET                 ; new PC=191A, new CS=64AA, flags=180 (TRAP!)
           CS:                    EXECUTION ROUTED BACK TO 64AA:1742
           MOV   BX,[09AC0]       FOR 29 TOTAL TIMES
            :
```

Routine to call on 30th ($1E_{th}$) iteration from 64AA:1818

```
64AA 1577   MOV   Byte ptr [0188],02
            TEST  Byte ptr [022C],04
            JNZ   159Ø
            MOV   AX,4Ø
            MOV   ES,AX        ; use ES to address relative to 00400H
            MOV   AL,ES:[003F] ; retriere Drive motor status (bits 0-3 = drive Ø-3) should be zero
            TEST  AL,3         ; A or B motor running?
            JNZ   159A         ; go if they are; otherwise...
            MOV   AH,19
            INT   21
            CMP   AL,Ø1
            JBE   159C         ; go if current drive is A or B (Ø or 1); otherwise
            MOV   AL,Ø1
159A        DEC   AL
159C        MOV   [0184],AL    ; save drive to check
159F        *CALL 1693
64AA 1693   MOV   AX,[0186]
            MOV   CL,04
            SHL   AX,CL
            ADD   AL,[0182]
            MOV   BL,[0182]
            SHR   BL,CL
            ADD   AH,BL
            MOV   [022E],AX
            *RET
64AA 15A2   *CALL 151E
64AA 151E   MOV   Byte ptr [23Ø],55
            MOV   Byte ptr [22C],04
            JZ    1537
            MOV   AH,02
            *CALL 15ØC
```

```
64AA 15ØC    MOV   AL,Ø1
             LEA   BX,[Ø742]
             PUSH  CS
             POP   ES
             MOV   DX,[Ø84]
             MOV   CX,1101
             JMP   ØA14
64AA ØA14    PUSH  BX
             PUSH  CX
             PUSH  DS
             PUSH  SI
             PUSH  DI
             PUSH  BP
             PUSH  DX
             TEST  Byte ptr [ØZZC],Ø3
             JNZ   ØA25
             JMP   ØDEB
     ØA25    MOV   BP,SP
             *CALL ØDE3
64AA ØDE3    PUSH  AX
             MOV   AX,4Ø
             MOV   DS,AX
             POP   AX
             *RET
     ØA2A    *CALL ØA47
64AA ØA47    MOV   CS:[Ø736],AH
             MOV   Word ptr CS:[Ø734],9ØØØ
             MOV   DH,AL
             AND   Byte ptr [ØØ3F],7F
             CMP   AH,1B
             JNZ   ØA62
             JMP   ØB14
```

```
64AA  0A62  OR   AH,AH
            JZ   0A8D
            DEC  AH
            JZ   0ADD
            MOV  Byte ptr[41],0     ; clear diskette status
            CMP  DL,04              ; DL should still have drive # (0..3)
            JNB  0A87
            DEC  AH
            JZ   0AE1
            DEC  AH
            JNZ  0A7F
            JMP  0B19
            DEC  AH
            JZ   0AEA
            DEC  AH
            JZ   0AEE
0A87        MOV  Byte ptr[41],01    ; Bad command
            RET
0A8D        MOV  DX,03F2            ; Digital output register (I/O)
            CLI
            MOV  AL,[3F]
            MOV  CL,04
            SHL  AL,CL
            TEST AL,20
            JNZ  0AA8
            TEST AL,40
            JNZ  0AA6
            TEST AL,80
            JZ   0AAA
            INC  AL
0AA6        INC  AL
0AA8        INC  AL
```

```
64AA 0AAA   OR   AL,08
            OUT  DX,AL
            MOV  Byte ptr [3E],0   ; clear seek status
            MOV  Byte ptr [41],0   ; clear drive status
            OR   AL,04
            OUT  DX,AL             ;enable FDC
            STI
           *CALL 0D12              ;
64AA 0D12  *CALL 0D33
64AA 0D33   STI
            PUSH BX
            PUSH CX
            CMP  Byte ptr cs:[0736],1B
            JNZ  0D47
            MOV  BL,01
            MOV  CX,cs:[0734]
            JMP  0D4B
      0D47  MOV  BL,02
            XOR  CX,CX
      0D4B  TEST Byte ptr [3E],80   ; waiting for interrupt
            JNE  0D5E
            LOOP 0D4B
            DEC  BL
            JNZ  0D4B
            OR   Byte ptr [41],80   ; time-out
            STC
      0D5E  PUSHF
            AND  Byte ptr [3E],7F   ; clear interrupt status
            POPF
            POP  CX
            POP  BX
           *RET
```

```
64AA  0D12   JB    0D2B    ; go if carry set (bad news - timeout?)
             MOV   AH,08     ; SENSE STATUS COMMAND
             *CALL 0C41
64AA  0C41   PUSH  DX
             PUSH  CX
             MOV   DX,03F4   ; FDC main status
             XOR   CX,CX
0C48         IN    AL,DX     ; get status
             TEST  AL,40     ; wait until FDC ready for command
             JZ    0C59
             LOOP  0C48
0C4F         OR    Byte ptr[41],80  ; flag timeout bit
             POP   CX
             POP   DX
             POP   AX
             STC
             RET
0C59         XOR   CX,CX
             IN    AL,DX
             TEST  AL,80
             JNZ   0C64      ; FDC data register ready
             LOOP  0C5B
             JMP   0C4F
0C64         MOV   AL,AH
             MOV   DL,F5     ; change to FDC data port
             OUT   DX,AL
             POP   CX
             POP   DX
             *RET
64AA  0D1C   JB    0D2B    ; go if carry set (bad)
             MOV   AL,[42]   ; 1st of 7 bytes that hold status bytes
             AND   AL,60     ;
```

```
64AA  0D26   CMP   AL,60
             JZ    0D2C
             CLC
             *RET
64AA  0ABE   MOV   AL,[42]   ; 1st status reg. byte
             CMP   AL,C0
```

Just prior to first diskette operation...

```
64AA  0B2F   PUSH   AX
             PUSH   CX
             MOV    CL,DL      ; DL = 0  (drive #?)
             MOV    AL,01
             SHL    AL,CL      ; AL = 1  (drive bit?)
             CLI
             MOV    Byte ptr[40],FF   ; DS = 40 (absolute 400) - set drive timeout counter
             TEST   [003F],AL  ; 43F = 0  (motor status bits)
             JNZ    0B74       ; so we don't jump
             AND    Byte ptr[3F],F0   ; clear all motor status bits
             OR     [3F],AL    ;    and or-in our drive bit
             STI
             MOV    AL,10
             SHL    AL,CL
             OR     AL,DL
             OR     AL,0C
             PUSH   DX
             MOV    DX,03F2    ; start your engines
             OUT    DX,AL
             POP    DX
             TEST   Byte ptr[3F],80 ;
             JZ     B74
             :
0B74         STI
             POP    CX
             CMP    Byte ptr cs:[736],18
             JZ     0B8F
             :
0B8F         POP    AX
             MOV    BH,AH
             MOV    DH,0
             JB     0BF0       ; jump if carry set
```

```
64AA  0B96    MOV     SI,01F0     ; address of error routine?
              PUSH    SI
      0B9A    CALL    0C41
                ---->
      0C41    PUSH    DX
              PUSH    CX
              MOV     DX,03F4     ; main status register
              XOR     CX,CX
      0C48    IN      AL,DX
              TEST    AL,40
              JZ      0C59        ; bit is ready
              LOOP    0C48
              OR      Byte ptr[41],80
              POP     CX
              POP     DX
              POP     AX
              STC
              RET
      0C59    XOR     CX,CX
      0C5B    IN      AL,DX
              TEST    AL,80
              JNZ     0C64        ; bit is ready
              LOOP    0C5B
              JMP     0C4F
              MOV     AL,AH       ; Function is 4A "READ ID" (in MF mode)
              MOV     DL,F5       ; switch ports
              OUT     DX,AL
              POP     CX
              POP     DX
              RET
                <----
      0B9D    MOV     AH,[BP+1]
```

```
64AA  0BA0   SUB   AH,1
             SHL   AH,1
             AND   AH,04
             OR    AH,DC
      0BA9   CALL  0C41   ; select HEAD 0, UNIT 0 for "READ ID" command
                  ⇄
      0BAC   CMP   BH,4D
             JNZ   0BB4   ; jump
             JMP   0AFC   ;
      0BB4   CMP   Byte ptr CS:[736],1B
             JNZ   0BC4   ; don't jump
             CMP   BH,4A  ; match?
             JNZ   0BC4
             JMP   0BEC
              ⋮
      0BEC   POP   SI
             CALL  0D33   ; wait for interrupt
                  ⇄
             JB    0C37   ; jump if carry set
      0BF2   CALL  0D7A
                  →
      0D7A   CLD
             MOV   DI,0042
             PUSH  CX
             PUSH  DX
             PUSH  BX
             MOV   BL,7   ; retrieve all 7 result bytes to 0042
      0D83   XOR   CX,CX
             MOV   DX,03F4 ; main status
      0D88   IN    AL,DX
             TEST  AL,80
             JNZ   0D99   ; bit is ready
```

```
64AA 0D8D   LOOP   0D88
             OR    Byte ptr [41],80
     0D94    STC
             POP   BX
             POP   DX
             POP   CX
             RET
     0D99    IN    AL,DX
             TEST  AL,40
             JNZ   0DA5     ; bit should be set (from FDC to processor)
     0D9E    OR    Byte ptr [41],20
             JMP   0D94
     0DA5    INC   DX       ; move to data port
             IN    AL,DX
             MOV   [DI],AL
             INC   DI
             MOV   CX,0A
     0DAD    LOOP  0DAD
             DEC   DX
             IN    AL,DX
             TEST  AL,10    ; FDC busy?
             JZ    0DBB     ; should jump after 7th byte read
             DEC   BL
             JNZ   0D83
             JMP   0D9E     ;
     0DBB    POP   BX
             POP   DX
             POP   CX
             RET
             <-
     0BF5    JB    0C36     ; jump on error
```

```
64AA  0BF7   CLD
             MOV    SI,0042   ; where results are stored
             LODSB
             AND    AL,C0     ; check interrupt code (should be 0)
             JZ     0C3B
              :
      0C3B   CALL   0DBF      ;
             ->
      0DBF   MOV    AL,[0045] ; get cylinder #
      0DC2   CMP    AL,CH     ; compare against 11h (AL = 07)
             MOV    AL,[0047] ; sector #
             JZ     0DD3
             MOV    BX,8
             CALL   0C6C
             MOV    AL,AH
             INC    AL
      0DD3   SUB    AL,CL
             RET
             <-
      0C3E   XOR    AH,AH
             RET               ; return to 0A2D, then to 153C, then to 15A5


      15A5   JB     15AC
      15A7   CALL   154B      ; we're off again, maybe now to actually read something
             ->
      154B   TEST   Byte ptr[22C],04   ; [22C] = 01
             JZ     1557      ; that bit is zero, so jump
             CALL   16AA
             JNB    1576
      1557   MOV    Word ptr [986],3
             CALL   14F2
             ->
```

```
6(AA 14F2   MOV   AH,02
            MOV   AL,01
            LEA   BX,[0742]
            PUSH  CS
            POP   ES
            MOV   CX,[0182]   ; CX= 2702   ;  1101
            MOV   DX,[0184]   ; DX= 0000   ;  0001
            JMP   0A14
0A14        PUSH  BX
            PUSH  CX
            PUSH  DS
            PUSH  SI
            PUSH  DI
            PUSH  BP
            PUSH  DX
            TEST  Byte ptr [022C],3
            JNZ   0A25        ;jump
            JMP   0DEB
0A25        MOV   BP,SP
0A27        CALL  0DE3
            →
0DE3        PUSH  AX
            MOV   AX,40
            MOV   DS,AX
            POP   AX
            RET
            ←
0A2A        CALL  0A47
            →
0A47        MOV   cs:[0736],AH      ; AH = 02
            MOV   Word ptr cs:[734],9000
            MOV   DH,AL             ; AL,DH = 01
```

```
64AA  0A55   AND    Byte ptr [3F],7F
             CMP    AH,1B
             JNZ    0A62          ; AH = 02
             JMP    0B14
0A62         OR     AH,AH
             JZ     0A8D          ; nope
             DEC    AH
             JZ     0ADD          ; nope
             MOV    Byte ptr [41],0
             CMP    DL,04         ; DL = 00
             JNB    0A87
             DEC    AH
             JZ     0AE1          ; go
              :
0AE1         MOV    AL,46         ; use DMA channel 2 - single byte transfers (Disk read)
0AE3         CALL   0CC8
             ---->
0CC8         PUSH   CX
             CLI
             OUT    0C,AL         ; reset first/last flip-flop
             PUSH   AX
             POP    AX
             OUT    0B,AL         ; output mode byte (4AH for write disk)
             MOV    AX,ES
             MOV    CL,04
             ROL    AX,CL
             MOV    CH,AL
             AND    AL,F0
             ADD    AX,BX
             JNB    0CE0
             INC    CH
0CE0         PUSH   AX
```

```
64AA  0CE1  OUT   04,AL    ; out with low 8-bits of address  (E2H)
            MOV   AL,AH
            OUT   04,AL    ; and high 8-bits                 (51H)
            MOV   AL,CH
            AND   AL,0F                                      (6H)
            OUT   81,AH    ; upper 4-bits (of full 20-bit address) to special DMAPAGE reg.
            MOV   AH,DH
            SUB   AL,AL
            SHR   AX,1
            PUSH  AX
            MOV   BX,6
0CF7        CALL  0C6C
            →
0C6C        PUSH  DS
            SUB   AX,AX
            MOV   DS,AX
            LDS   SI,[0078]
            SHR   BX,1
            MOV   AH,[BX+SI] ; AH = 4th byte in diskette control table
            POP   DS
            JB    0C41
            RET
            ←
0CFA        MOV   CL,AH      ; Bytes-per-sector = code 2 = 512 bytes/sector
            POP   AX
            SHL   AX,CL
            DEC   AX
0D00        PUSH  AX
            OUT   05,AL      ; byte-count      (FFH)
            MOV   AL,AH
            OUT   05,AL                        (01H)
            STI
```

absolute DMA address = 651E2H
relative to 64AA = 64AA:0742
(holds 1 512-byte sector) to
64AA:0941

Total to transfer:
512 bytes

```
64AA  00C8    POP     CX
              POP     AX
              ADD     AX,CX
              POP     CX
              MOV     AL,02
              OUT     0A,AL       ; enable DMA channel 2
              RET
              ←
      00E6    MOV     AH,E6
              JMP     0B25
              :
      0B25    JNB     0B2F        ; jump ('cause no carry)
      0B27    MOV     Byte ptr [41],09
              MOV     AL,0
              RET
      0B2F    PUSH    AX          ; AH = E6, AL = 02
              PUSH    CX          ; CX = 2702
              MOV     CL,DL       ; both CL & DL = 0  (drive #)
              MOV     AL,01
              SHL     AL,CL       ; AL = 01
              CLI
              MOV     Byte ptr[40],FF ; Drive motor timeout
              TEST    [3F],AL     ; is motor on?
              JNZ     0B54
              AND     Byte ptr[3F],F0 ; probably time to make motor go on
              OR      [3F],AL
              STI
              MOV     AL,10
              SHL     AL,CL
              OR      AL,DL
              OR      AL,0C
              PUSH    DX
```

```
64AA  0B56   MOV    DX,03F2      ; digital output register
             OUT    DX,AL
             POP    DX
             TEST   Byte ptr [0F],80
             JZ     0B74
             MOV    BX,0014      ; get 11th (20 DIV 2 + 1) byte in diskette table
             CALL   0C6C         ; for how long to allow for start-up in AH
             <=>
             OR     AH,AH
0B6A         JZ     0B74         ; should be ready by now
             SUB    CX,CX
0B6E         LOOP   0B6E
             DEC    AH
             JMP    0B6A
0B74         STI
             POP    CX
             CMP    Byte ptr cs:[736],1B   ; byte = 02
             JZ     0B8F
             MOV    word ptr cs:[734],0
0B85         CALL   0C7D
             ->
0C7D         MOV    AL,01
             PUSH   CX
             MOV    CL,DL        ; DL = drive #
             ROL    AL,CL
             POP    CX
             TEST   [3E],AL      ; bit should be non-zero if no recal. needed
             JNZ    0C9E
             MOV    AH,07        ; do "Recalibrate"
             CALL   0C41
             <=>
             MOV    AH,DL
```

```
64AA  ØC94    MOV   AH,DL      ; 2nd byte of command = drive #
              CALL  ØC41
                  ⇒
              ⇐
      ØC99    CALL  ØD12
                  →
      ØD12    CALL  ØD33      ; NO DETAIL HERE - wait for interrupt
              JB    ØD2B
              MOV   AH,Ø8
              CALL  ØC41      ; NO DETAIL HERE - ISSUE "SENSE INT" command
              CALL  ØD7A      ; NO DETAIL HERE - read all result bytes
              JB    ØD2B
              MOV   AL,[42]   ; 1st result byte
              AND   AL,6Ø
              CMP   AL,6Ø
              JZ    ØD2C
              CLC
              RET
      ØD2C    OR    Byte ptr[41],4Ø
              STC
              RET
                  ←
      ØC9C    JB    ØCC7
              MOV   AH,ØF     ; "Seek" command
              CALL  ØC41
              MOV   AH,DL     ; 2nd byte of command = drive #
              CALL  ØC41
              MOV   AH,CH     ; 3rd byte = NCN (new cyl. # = 27H or 39!)
              CALL  ØD12      ; wait for interrupt, get results, set carry (clear hopefully)
              PUSHF
              MOV   BX,12
              CALL  ØC6C      ; get head settle time
      ØCB7    PUSH  CX
```

```
64AA  0CB8   MOV   CX,0226
             OR    AH,AH      ; head settle time - 0F
             JZ    0CC5
      0CBF   LOOP  0CBF
             DEC   AH
             JMP   0CB8
             POP   CX         ; seek command complete - restore carry
             POPF
             RET
              ←

      0B88   MOV   word ptr cs:[734],9000
             POP   AX
             MOV   BH,AH
             MOV   DH,0
             JB    0BF0
             MOV   SI,01F0
             PUSH  SI
             CALL  0C41       ; AH = E6 for "Read Data - multi-track, and SK"
             MOV   AH,[BP+01]
             SHL   AH,1
             SHL   AH,1
             AND   AH,04
             OR    AH,DL
             CALL  0C41       ; 2nd byte = drive #
             CMP   BH,4D
             JNZ   0BB4       ; jump
             JMP   0AFC
             CMP   byte ptr cs:[736],1B
             JNZ   0BC4       ; jump
             CMP   BH,4A
             JNZ   0BC4       ; jump - its E6 this time
             JMP   0BEC
             NOP
```

```
64AA  0BC4    MOV     AH,CH
              CALL    0C41    ; 3rd byte = cyl # = 39.
              MOV     AH,[BP+1]
              CALL    0C41    ; 4th byte = head
              MOV     AH,CL
              CALL    0C41    ; 5th byte = sector = 2.
              MOV     BX,7
              CALL    0C6C    ; 6th byte = no. bytes per sector = 512 (code 2)
              MOV     BX,9
              CALL    0C6C    ; 7th byte = end-of-track
              MOV     BX,B
              CALL    0C6C    ; 8th byte = GPL between sectors
              MOV     BX,D
              CALL    0C6C    ; 9th byte = data length
              POP     SI
              CALL    0033    ; wait for interrupt
              JB      0C37
              CALL    007A    ; read all result bytes
              JB      0C36
              CLD
              MOV     SI,0042
              LODSB
              AND     AL,C0
              JZ      0C3B    ; jump if ok
              :
0C3B          CALL    0DBF    ; get # of sectors read in AL
              XOR     AH,AH
              RET
              ←
0A2D          MOV     BX,0004
              CALL    0C6C    ; get wait time to turn motor off
```

```
64AA  0A30    CALL    0C6C
              MOV     [40], AH
              MOV     AH, [41]
              CMP     AH, 01
              CMC
              POP     DX
              POP     BP
              POP     DI
              POP     SI
              POP     DS
              POP     CX
              POP     BX
              RET
              <-
      1560    JNB     1576    ; READ DATA operation complete
              TEST    AH, 10  ; Bad CRC - no
              STC
              JNZ     1576    ; jump if bad crc
              TEST    Byte ptr [220], 4
              JNZ     1575    ; don't jump
              DEC     word ptr [486]
              JNZ     155D    ; 3 retries
              CLC
      1576    RET
              <-
->    15AA    JB      15C6
              DEC     Byte ptr [488]      ; goes to 1
              XOR     Byte ptr [184], 01  ; low bit goes to 1
              CALL    151E
```

```
64AA  15AA   JB      15C6        ; we should jump
              :
      15C6   MOV     AX,[4F2]    ; get first word sector (DB2C)
             MOV     [0989],AX   ; save it
             MOV     AL,AA
             MOV     CX,200      ; CX = 512.
  →  15D1    PUSH    CS
             POP     ES
             MOV     DI,0742
             REPE
             STOSB               ; now wipe out entire sector in memory
             MOV     word ptr[986],2
      15DE   CALL    15D8
             →
             MOV     AH,03
             JMP     14F4
             (Do another disk operation)
             ←
  →  15E1    JNB     15F9
```