DazzleStar Disassembler:

DZ.COM uses RST 2, RST 3 and RST 4 instructions for fast jumps to keep the screen updated:

RST 2:(0010H)	**RST 3:(0018H)**	**RST 4:(0020H)**
PUSH IX	JMP 8C87H	JMP 8CA6H
POP D		
DAD D		
RET		

For Heath computers, this won't work since RST 3 is already in use for the keyboard interrupt. The solution appears to be to use RST 5 (0028H) instead.

When executed, an RST instruction will push the PC to the stack and then execute the code at the RST address. RST 2 will execute and return to the instruction following the RST 2 command. RST 3 and 4 will jump to the specified address and branch according to the program code and conditions and return to the instruction following the RST n instruction when a RET pops the value pushed when the RST was encountered.

When first started, the first command is a jump to the routine to test the program and equipment status. The CPU must be a Z-80. The program will check the code by performing a simple checksum of all bytes between 0300H and 8FB6H and comparing the result to the stored value at 8FB7 and 8FB8. It then checks location of the BDOS to determine if the TPA top is less than 38400k. It will exit to CP/M if any test fails. If all checks are good, the program will relocate almost all code and set up the program's work space around 9000H to the top of TPA.

Revising the program to use RST 5 instead of RST 3 will change the checksum and the stored value must be updated to the new sum. This can be accomplished by appending “CKSUM.HEX”. This will read the modified DZ-HEATH.COM (DZ.COM configured for Heath H-19 terminal) into memory, compute the new checksum, replace the stored checksum value and exit to CP/M. The revised program is still in memory and can be saved with the command “SAVE 143 filename.ext”.

CKSUM.Z80

(Assemble to a HEX file for appending with DDTZ.COM.)

```
; Code to be added to the end of the revised DZ using DDTZ.
; A jump to this routine will perform the simple
; sixteen-bit sum of all bytes between 0300h and 8FB6h and
; will install at location 8FB7h of the revised program.
; Immediately 'save 143 filename.ext' or peek the memory
; location and edit the file to change the value.
;

        .Z80
        ASEG
        ORG   9000H
```

```
        LD      B,00H           ;clear registers for use
        LD      HL,0000
        LD      DE,0103H        ;start address in DE
LOOP:   EX      DE,HL
        LD      C,(HL)          ;load byte to C
        INC     HL              ;inc to next byte
        EX      DE,HL           ;address in DE, sum in HL
        ADD     HL,BC           ;add byte to sum
        LD      A,07FH                  ;end low byte to A
        CP      E               ;match?
        JR      NZ,LOOP         ;no
                ;
        LD      A,8AH           ;end high byte to A
        CP      D               ;on last page?
        JR      NZ,LOOP         ;no
                ;
        EX      DE,HL           ;HL is now at B7h
        LD      (HL),E          ;write low byte
        INC     HL
        LD      (HL),D          ;write high byte
        CALL    0000            ;exit to CP/M
        END
```

Using the ZCPR 3.4 (NZCOM) intrinsic PEEK command will display memory in half page blocks. At the CP/M prompt, typing P 0100 will display half of the 01 page. Typing P will display the following half page, etc.

What I have found is that the program relocates most of the code to areas other than where is resides when loaded for disassembly. If we run the program and immediately exit back to CP/M, we can save the memory contents of the 'run-time' environment to a file. This can now be disassembled to see how the program actually works. By generating a relocation map for the 'DF' bytes, the original file can be loaded and patched where needed.

The relocation code follows:

DazzleStar -- Program Code Relocation (Ancient form of copy protection!)

```
                ; Routine for Code Moves
                ;
8EEE 4E     J$8EEE:     LD    C,(HL)      ; Source in BC
8EEF 23     I$8EEF:     INC   HL
8EF0 46     D.8EF0:     LD    B,(HL)
8EF1 23                 INC   HL          ; pointer to next data
                ;
                ;       Subroutine _____________________________
                ;         Inputs   _____________________________
                ;         Outputs  _____________________________
                ;
8EF2 C5     C$8EF2:     PUSH  BC          ; BC to stack
8EF3 5E                 LD    E,(HL)      ; LD DE
8EF4 23                 INC   HL
8EF5 56                 LD    D,(HL)
8EF6 23                 INC   HL          ; pointer to next data
8EF7 AF                 XOR   A
8EF8 BB                 CP    E
8EF9 2006               JR    NZ,J.8F01   ; zero when moves done
                ;
8EFB BA                 CP    D
8EFC 2003               JR    NZ,J.8F01
                ;
8EFE E1                 POP   HL
8EFF D5                 PUSH  DE
8F00 E9                 JP    (HL)        ; start running program
                ;
                ;       ----------------
8F01 4E     J.8F01:     LD    C,(HL)      ; LD BC
8F02 23                 INC   HL
8F03 46                 LD    B,(HL)
8F04 23                 INC   HL          ; HL = next moves data
8F05 E3                 EX    (SP),HL     ; source in HL, move addr ptr on stack
8F06 EDB8               LDDR
8F08 E1                 POP   HL          ; get saved move addr pointer in HL
8F09 18E3               JR    J$8EEE
                ;
                ; Data for moves 2 through xx using LDDR
                ; S = Source, D = Destination, C = Count
                ;       ----------------          Usage
8F0B 7B8E   ?.8F0B:     DEFW  8E7BH       S
8F0D FF93               DEFW  93FFH       D
8F0F 0100               DEFW  0001H       C
8F11 FF93               DEFW  93FFH       S
8F13 FE93               DEFW  93FEH       D
8F15 BC00               DEFW  00BCH       C
8F17 7A8E               DEFW  8E7AH       S
```

8F19 4293	DEFW 9342H	D
8F1B 8101	DEFW 0181H	C
8F1D C291	DEFW 91C2H	S
8F1F C191	DEFW 91C1H	D
8F21 0B00	DEFW 000BH	C
8F23 F98C	DEFW 8CF9H	S
8F25 B691	DEFW 91B6H	D
8F27 4303	DEFW 0343H	C
8F29 748E	DEFW 8E74H	S
8F2B 738E	DEFW 8E73H	D
8F2D 6200	DEFW 0062H	C
8F2F B689	DEFW 89B6H	S
8F31 118E	DEFW 8E11H	D
8F33 B10D	DEFW 0DB1H	C
8F35 6180	DEFW 8061H	S
8F37 6080	DEFW 8060H	D
8F39 1100	DEFW 0011H	C
8F3B 057C	DEFW 7C05H	S
8F3D 4F80	DEFW 804FH	D
8F3F 140A	DEFW 0A14H	C
8F41 3C76	DEFW 763CH	S
8F43 3B76	DEFW 763BH	D
8F45 0F00	DEFW 000FH	C
8F47 F171	DEFW 71F1H	S
8F49 2C76	DEFW 762CH	D
8F4B 9D02	DEFW 029DH	C
8F4D 9073	DEFW 7390H	S
8F4F 8F73	DEFW 738FH	D
8F51 1400	DEFW 0014H	C
8F53 546F	DEFW 6F54H	S
8F55 7B73	DEFW 737BH	D
8F57 2904	DEFW 0429H	C
8F59 536F	DEFW 6F53H	S
8F5B 526F	DEFW 6F52H	D
8F5D 0C00	DEFW 000CH	C
8F5F 2B6B	DEFW 6B2BH	S
8F61 466F	DEFW 6F46H	D
8F63 3401	DEFW 0134H	C
8F65 136E	DEFW 6E13H	S
8F67 126E	DEFW 6E12H	D
8F69 2500	DEFW 0025H	C
8F6B F769	DEFW 69F7H	S
8F6D ED6D	DEFW 6DEDH	D
8F6F 4312	DEFW 1243H	C
8F71 AB5B	DEFW 5BABH	S
8F73 AA5B	DEFW 5BAAH	D
8F75 3B00	DEFW 003BH	C
8F77 B457	DEFW 57B4H	S
8F79 6F5B	DEFW 5B6FH	D

```
8F7B 0307          DEFW 0703H      C
8F7D 6D54          DEFW 546DH      S
8F7F 6C54          DEFW 546CH      D
8F81 0B00          DEFW 000BH      C
8F83 B150          DEFW 50B1H      S
8F85 6154          DEFW 5461H      D
8F87 3522          DEFW 2235H      C
8F89 2D32          DEFW 322DH      S
8F8B 2C32          DEFW 322CH      D
8F8D C400          DEFW 00C4H      C
8F8F 7C2E          DEFW 2E7CH      S
8F91 6831          DEFW 3168H      D
8F93 0D14          DEFW 140DH      C
8F95 5C1D          DEFW 1D5CH      S
8F97 5B1D          DEFW 1D5BH      D
8F99 5201          DEFW 0152H      C
8F9B 6F1A          DEFW 1A6FH      S
8F9D 091C          DEFW 1C09H      D
8F9F 380B          DEFW 0B38H      C
8FA1 D210          DEFW 10D2H      S
8FA3 D110          DEFW 10D1H      D
8FA5 8600          DEFW 0086H      C
8FA7 370F          DEFW 0F37H      S
8FA9 4B10          DEFW 104BH      D
8FAB 390C          DEFW 0C39H      C
8FAD 1304          DEFW 0413H      S
8FAF 1204          DEFW 0412H      D
8FB1 1401          DEFW 0114H      C
8FB3 494D          DEFW 4D49H      S       ; address to jump to after moves
8FB5 0000          DEFW 0000H      D       ; this will end the moves
8FB7 62E1          DEFW 0E162H
8FB9 00000000      DEFB 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

From the Relocation code above, a Relocation Table was made for all of the 'DF' bytes.

In addition to changing the RST 3 instructions to RST 5, we must modify the code to setup the RST 5 vector instead of the RST 3 vector.

```
;Code to setup RST routines
;
----------------
X.8800:         LD      HL,(D$8C82)
                LD      (X.0010),HL             ; RST 2
                LD      HL,(D$8C84)
C.8807 EQU      $-2
                LD      (D.0012),HL
                LD      HL,(D$8C86)
C$880E EQU      $-1
                LD      (D.0014),HL
                LD      A,0C3H                  ; JMP instruction
                LD      (X.0028),A              ; install JMP at RST 5 (revised)
C$8815 EQU      $-2
                LD      (X.0020),A              ; install JMP at RST 4
                LD      HL,I$8C87               ; jump address for RST 5
                LD      (D$0029),HL             ; put it in RST 5 vector (revised)
                LD      HL,I$8CA6               ; jump address for RST 4
C$8823:         LD      (D.0021),HL             ; put it in RST 4 vector
                RET
```

With the changes outlined above, the program appears to run on Heath computers with a Z80 processor. Testing is still in progress to determine if additional changes are required.