DazzleStar – No RSTs Version

DazzleStar © November 1986 was written by Mr. John Washington and released into the CP/M world via 'DZ-NOV86.LBR'. The released version of DZ.COM uses RST 2, RST 3 and RST 4 instructions for fast jumps to keep the multiple screen areas updated to allow an interactive disassembly environment and to reduce the program size to conserve the precious TPA on systems running CP/M 2. While the Heath H-19 terminal is listed in the install menu, the program will not run on Heath computers since RST 3 is already in use as the keyboard interrupt. When loaded, it would start and run normally up to the point that it overwrote the keyboard interrupt. At this point, you could no longer type into the program.

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

I worked for years trying to work around this conflict by using a different RST vector – for example use RST 5 instead of RST 3. The problem is that Heath uses RST 5 as part of the H-37 soft-sectored floppy disk service routine. The program worked great on CP/M machines that met the minimum requirements – any CP/M environment where the CPU is a Z80 (or equivalent), the code passes the sum of bytes test, TPA space is sufficient **AND the RST vectors are not used by the OS and are available for use.**

I finally admitted defeat and sought some expert help from Mr. Douglas Miller. Once he explored the program, he was impressed with how the program worked and agreed to help me get it so we could use on our Heath computers.

His efforts have been two fold:

1. He ported the code into a Java based program that can be run on the more recently developed machines running Windows or Linux.
2. He reworked the code to run without dependence on the RST jumps so that we can run it on our Heath computers.

One of the things that I learned about the program while attempting to use RST5 was that Mr. Washington was an excellent programmer and was well versed in the Z-80 command set. I believe he used almost all of the op codes! Another was that he used subroutines that could be called for multiple services by passing information via one or two bytes at the return address on the stack. When these routines are called, they will exchange SP and HL registers, read the bytes passed, and then, with HL pointing to the byte after the last byte read, again exchange SP and HL. This means the RETURN command will return to the byte after the passed data, and HL is back to the state when the routine was called. This technique was used for RST 3 and RST4 and several of the called subroutines.

In order to maintain the code as close as possible to the original code done by Mr. Washington, we still have to pass these bytes. The RST instructions were an efficient way of calling frequently used service routines and can be replaced by a standard call instruction at the expense of making the program a few bytes larger. The CALL commands that pass data are:

CALL 9379 replaces the original RST2 (0010 h, {D7 h}).
CALL 937E replaces the original RST3 (0018 h, {DF h}) passes 1 byte to the service routine.
CALL 939D replaces the original RST4 (0020 h, {E7 h}) passes 1 byte to the service routine.

CALL 7C87 This CALL passes 1 byte to the called routine. (originally 7569 h)
CALL 9358 This CALL passes 1 byte to the called routine. (originally 8C3A h)
CALL 936B This CALL passes 2 bytes to the called routine. (originally 8D71 h)
CALL 941E This CALL passes 2 bytes to the called routine. (originally 8D27 h)

Otherwise, the program code is unchanged and works exactly the same as the originally distributed file.

Likewise, the DZINSTAL.COM file was also reworked to install any of the listed terminals into the reworked DZ.COM file.

Included with the new files you will find DZ-NORST.LST. This is a rough disassembly of the new file with the passed bytes marked for the CALL instructions that pass bytes, listed above.