August 2012

Z-100 LifeLine

Since 1989

#122

A Professional Journal Exclusively for the Heath/Zenith Z-100 Computer

Z-100 LifeLine Web Site: **https://z100lifeline.swvagts.com (new effective September 2019)**

HOWGOZIT

Well, the summer is screaming by and it's proving to be just as hot as I was concerned about. So while the heat is baking everything outside - except the weeds, I've decided to delve back into the books to get a handle on CP/M, a strange operating system from the extreme past.

I'm sorry to say that the last time I messed with CP/M I was converting my Paint Program from the H-89 to the new Z-100, over 25 years ago! So please bear with me if I am kind of slow in this process or I mess up while explaining an otherwise simple concept.

What got me interested in this old operating system? Well, a promise that we would look into getting our new Z100LL IDE Controller card to work under at least one CP/M version someday. I've been putting it off, hoping that some software wiz would give me a call asking to help out... but I haven't heard from a soul, so I guess it is up to me.

And I certainly didn't want to work outside this summer - though the pool has been great, especially while Myra was home recovering from her hip replacement. But she's gone back to work now, so it was also time for me to get back to business.

Most of you probably haven't dealt much with CP/M, and then it was probably more years ago than you would wish to remember. However, after looking on the internet, it would appear that there is a lot more interest in CP/M than the aging DOS. Don't believe it?

Next time you are on the internet, take a peek at s100computers.com sometime. I don't know about the software side, but it appears that they have interested parties from all the aging computer groups of the early to mid-80's, tearing into and modifying their computers to do marvelous things. Every year or so they come out with a new circuit board to make things run faster or more efficiently and add more capability, such as a generic IDE controller, RTC clocks, RAM boards, and the like. Based on the more capable Z80 CPU, they have computers running at greater than 20MHz! Try that in our Z-100.

Personally, my next project was to learn more about DOS assembly programming so I could make some much needed modifications to some of our software. For example, in this issue I was going to discuss disassembling ZDIR.COM for the practice. But that project has just a few more hurdles to cross before it is ready for prime time.

I also thought that it may make life a bit easier if I re-learned the simpler CP/M programming concepts before tackling the more ambitious projects, so here we are.

So, for anyone interested, let's take a leap back a few years and tackle this project together. We might even enjoy the trip.

Before we begin, I need to bring you back up to speed on installing a CP/M partition on a hard drive. You can do my exercises on a dual floppy system and even a single floppy drive, but a single drive would be a royal pain.

Running CP/M on the H/Z-100

Hopefully, the last time you PREPped and PARTed your hard drive, you created a CP/M partition of some reasonable size, about 1Mb should be big enough for our purposes. I'm not going into specifics, as this process has been around forever and is adequately described in the manuals and even some back issues of the LifeLine. However, if you have not left a CP/M partition on your hard drive, a word of warning:

WARNING: Unless you already have a separate partition, maybe used for word processing, or spreadsheets, that you can rename, using PART to create a new CP/M partition will destroy all other data on the hard drive because this process will reduce the size of your other partition(s). Any partitions changed, moved or created will require reformatting!

There are several versions of CP/M that were used on the H/Z-100. The most popular were:

CP/M-80: An 8-bit system used on the H/Z-89 and transferred to the H/Z-100.

CP/M-85, or **CP/M v2.2:** A combined 8-bit & 16-bit version designed specifically to take advantage of the H/Z-100's more capable 8088 CPU.

CP/M Plus, more popularly known as **CP/M 3:** An enhanced version of CP/M-80 by Digital Research that added RAM banking to take advantage of the increased memory in more modern computers of the time, auto density drive select, device reassignment and a date/time capability for date-stamping files. CP/M Plus was modified and distributed by Barry Watzman to take maximum advantage of the H/Z-100's capabilities.

CP/M-86, MP/M-86, Concurrent CP/M-86, or **Concurrent DOS:** A version, or versions, with several names that was designed specifically for multiple users and allowed several terminals to be used simultaneously. It also featured multi-tasking and virtual consoles. Most notably, it had the ability to run a number of DOS and CP/M application programs, as well as to read/write DOS and CP/M files.

Like with DOS at the time, every computer had its own version of CP/M, making life very difficult. However, except for the system modules, most were very similar.

Initially, I loaded CP/M-85 onto my hard drive, but I later found that the software being used for the generic S-100 buss IDE controller board was for CP/M 3. So, that is the operating system we'll be using.

A Description of CP/M Plus (Commonly Known as CP/M 3)

Digital Research developed the CP/M Plus Operating System to take advantage of the latest hardware in the 8-bit microcomputer world, including greater memory and real time clocks. The user could use a simple non-banked RAM system or could install a banked RAM system with bank switching, auto density drive select, file date-time stamping and device reassignment.

Barry A. Watzman then modified this further for use specifically on the H/Z-100. This is the release we will use for our project.

Note: If you do not have this version, if you send me a check for $8.00 made out to Steven W. Vagts, I'll be happy to send you a copy of the DSDD three-disk (5") set, with specific installation instructions. Sorry, I'm not set up to do 8" at the moment. I also cannot include the manuals, as I've only got the one set. However, for an additional $5.00 I can include a CP/M-85 manual, less binders.

CP/M Plus has some faults that I found irritating and worth further comment.

First off, the ASSIGN command, which is used to assign drive letters to CP/M hard drive partitions, is terrible compared to that of CP/M-85. ASSIGN ? does not work, so expect no help, and the errors are somewhat cryptic. But worse, and in spite of the way ASSIGN works in prior versions and ASGNPART in DOS, this ASSIGN will also not successfully assign a drive letter to a freshly PREPped, PARTitioned, and unformatted hard drive partition! FORMAT now does this when it runs on a hard drive partition. Fortunately, we don't need to mess with this very often, but I was stumped for quite a while trying to figure out why, and finally, I just used the older CP/M-85 to prepare the hard drive.

Second, several of the utilities are not Y2K compliant. You remember that exciting time approaching the year 2000? Well, dates are NOT accepted after 1999. The good news is that I found the patch for DATE.COM on the web, and the folks at s100computers.com provided me with working patched copies of DATE, SHOW, DIR and SETDEF. We'll do the patch for DATE later.

As a warmup exercise into CP/M assembly language programming, this issue's insert is a program that was created while taking the course *"Programming in 8080/8085 Assembly Language"* from Heathkit/Zenith Educational Systems, 1982 edition. You may recall the All-Base Conversion Program from when the

H/Z-89 ruled - way back. I've updated the routines to take advantage of the Z-100's BDOS calls, removed the Split-Octal routine, and added a Binary Coded Decimal display and the display of the graphic characters on the Z-100.

Note: This program and the four patched programs mentioned above have been added to the CP/M 3 distribution disks.

Now, we have to address another problem. Having become the de facto standard in moderm computers, DOS applications are freely exchanged over the web, can be easily down-loaded, and transfer readily by diskette among computers. Not so for CP/M.

Here's a question for you. How come there is a program, RDCPM, that can read CP/M data disks and put them on a DOS diskette, but not one that can read a file on a DOS diskette and place it on a CP/M diskette? Has there never been a need for such a program?

Anyway, the folks at s100computers.com do all their work under Windows on a PC - their website's CP/M files are all downloadable and treated as DOS files. Then they recommend using a null modem cable to connect a PC with a CP/M computer to transfer the files. They also use a CP/M simulator, which is available on their website, on a PC to test out their new programs and utilities. However, while the CP/M simulator simulates over 20 different computer systems of the 70's and 80's, the H/Z-89 and H/Z-100 were not included. If one of you feels up to tackling such a project, it would be greatly appreciated.

But, rather than connecting computers together by cable (I haven't got the space for all that), I found a program on the internet, 22DISK, that does what I need...

22DISK
CP/M <-> DOS Disk Interchange Utility

22DISK is a utility developed and licensed from Sydex, Inc, which still operates in Oregon, but is now providing disk salvage/ transfer services and no longer provides free downloads of its software - a long story we won't go into here. However, the fully functional 22DISK is still available from elsewhere. I can provide a copy if needed.

22DISK is loaded and run from a DOS window on any older PC clone and will require a 360K, 5.25" drive installed. As I already had one for my Z-100 work running under Windows 98, this was easy. I do not believe that this program will work under a later computer or operating system (my WinXP computers don't like 5.25" drives and don't have a DOS prompt capability anyway, though I suppose there are ways around both issues).

Originally, I had the ZIPped copy of 22Disk buried several directories down on my PC's hard drive, but then double-clicking on 22Dinst.BAT as the instructions desired brought up an MS-DOS window with an error stating that I had to be logged into the drive that I was to run the program on.

I finally moved all files of the program to a separate \22Disk directory on my work drive, E:, and double-clicked on the file again, but again it gave me the same error.

The only way that I found around it was to enter a DOS window, change the default drive from C:\Windows to E:\22Disk and then type in the command:

```
E:\22Disk>22Dinst E:\MY22
```

This seemed to work, but only placed the file MY22 in the E: drive's root directory. I moved it into the 22Disk directory with the other files.

From within the E:\22Disk directory, I double-clicked the CMENU.EXE file and this time I got CMenu's menu display in a DOS window.

```
+--------------------------------------+
|  22DISK Version 1.44 (Oct 31, 1996)  |
|                                      |
| Please select one of the following:  |
|                                      |
|   0.  Exit to DOS                    |
|   1.  Set CP/M diskette type         |
|   2.  Set CP/M diskette drive        |
|   3.  Copy CP/M file(s) to DOS       |
|   4.  Copy DOS file(s) to CP/M       |
|   5.  Format a CP/M diskette         |
|   6.  Display a CP/M directory       |
|   7.  Display (type) CP/M file(s)    |
|   8.  Erase CP/M file(s)             |
|                                      |
+--------------------------------------+
No diskette type selected yet.

  Please enter function number (0...8):_
```

Pressing **1** gave me the statement:

```
  Enter 1-4 character disk format type,
  or ENTER alone for list -
```

As this was the first time through, I pressed **ENTER**. A listing of all kinds of computer brands and format types appeared in a DOS Window.

of interest to us Heath/Zenith users were:

```
A1     Generic CP/M - SSSD 8"
EPS3   Epson PX-8 - DSDD 3.5"
HEA1   Heath H89, Magnolia CP/M - SSDD 48 tpi 5.25"
HEA3   Heath H89, Magnolia CP/M - DSDD 96 tpi 5.25"
IBM1   IBM PC, CP/M-86 - SSDD 48 tpi 5.25"
IBM2   IBM PC, CP/M-86 - DSDD 48 tpi 5.25"
NEC4   NEC PC 8500/8431A, Starlet - DSDD 3.5"
OLI3   Olivetti 250 - SSDD 3.5"
SIE3   Siemens PG-635 DSDD 3.5"
ZEN1   Zenith Z-37 Disk - SSSD 48 tpi 5.25"
ZEN2   Zenith Z-37 Disk - DSDD 96 tpi 5.25"
ZEN3   Zenith Z89, Heath H89 - DSDD 48 tpi 5.25"
ZEN4   Zenith Z89, Heath H89 - DSDD 96 tpi 5.25"
ZEN5   Zenith Z90 - SSDD 48 tpi 5.25"
ZEN6   Zenith Z90 - DSDD 48 tpi 5.25"
ZEN7   Zenith Z-100 - SSDD 48 tpi 5.25"
ZEN8   Zenith Z-100 - DSDD 48 tpi 5.25"
```

I chose this last entry and pressed **ENTER** to select it.

The program responded with the following paragraph:

```
This format will work best on a standard
360K 5.25" diskette drive. If you are
using a PC-AT-type computer with only
1.2M high-density drives, you will have
no problem reading your CP/M diskettes,
but writing will be unreliable. If you
need to write diskettes using the high-
density drive, first bulk-erase them
before formatting them in the high-density
drive.

Press any key to continue...
```

This brought us back to the orignal CMenu screen, but with the statement:

```
ZEN8   Zenith Z-100 - DSDD 48 tpi 5.25",
no drive selected
```

printed immediately under the box, and the statement:

```
Please enter function number (0...8):_
```

This time entering **2**, the computer cleared the DOS Window and displayed:

```
SET CP/M DRIVE NAME

Enter CP/M drive name or unit (e.g., A:)
```

On my PC, my 5.25" DSDD drive is B:, so I responded with **B**. The computer displayed:

```
Press any key to continue...
```

We were back to the CMenu screen, but this time the statement was updated below the box with:

```
ZEN8   Zenith Z-100 - DSDD 48 tpi 5.25",
drive B:

Please enter function number (0...8):
```

I was anxious to format a new CP/M diskette, so I entered 5. The computer responded with:

```
FORMAT A CP/M DISKETTE

CFMT Ver. 1.44-Oct 31 1996, Copyright
 1996, Sydex. All rightsreserved.
 THIS IS AN UNREGISTERED COPY --
 SEE DOCUMENTATIONFOR DETAILS.
Zenith Z-100 - DSDD 48 tpi 5.25" format

Insert blank diskette into drive B: and
enter "G" to begin; Anything else will
exit to DOS without formatting - _
```

Pressing "G", we were on our way...

```
Formatting track xx (xx stopped at 39)
Format Complete - No Errors

Replace diskette in drive B: and type
any key to return to DOS
```

I had three updated CP/M files from s100computers.com on drive E: (my work hard drive) on my PC so I thought I would try copying from there first. Pressing 4 to Copy, the program responded with:

```
COPY DOS FILES TO CP/M

Enter DOS file name (e.g., C:\MYDIR\*.TXT
```

I entered E:\DATETIME\*.* and pressed ENTER. The program responded with:

```
Enter CP/M destination name. If you
specify a diskette drive (user number
is optional), it will override the
selected CP/M drive. If you do not
specify a name, the DOS filenames
will be used:
```

I entered **B** thinking it would at least need a destination drive letter and the computer responded with:

```
DTOC Ver. 1.44-Oct 31 1996, Copyright
1996, Sydex. All rights reserved.
THIS IS AN UNREGISTERED COPY --
SEE DOCUMENTATION FOR DETAILS.
Zenith Z-100 - DSDD 48 tpi 5.25" format

Copying E:\DATETIME\DATE.COM to B0:B
300K remaining on drive B:
Press any key to continue...
```

OK! Now let's try using the wildcards again... Well, maybe. There are three files I wished to copy: DATE.COM, DIR.COM and SHOW.COM. I entered: E:\DATETIME\*.*, and all the computer responded with was:

```
   Copying E:\DATETIME\DIR.COM to B0:B

   288K remaining on drive B:
   Press any key to continue...
```

Back to the CMenu. Let's try **6** for "Display a CP/M directory".

```
   DISPLAY A CP/M DIRECTORY

   Enter CP/M file name. If you specify a
   diskette drive (user number is optional),
   it will override the selected CP/M drive.
   You may include 'wild card' characters
   (* and ?) in the name. ENTER will cause
   all files to be displayed:
```

Well, I wanted to see all the files, so I pressed **ENTER** alone.

```
   CDIR Ver. 1.44-Oct 31 1996, Copyright
   1996, Sydex. All rights reserved.
   THIS IS AN UNREGISTERED COPY --
   SEE DOCUMENTATION FOR DETAILS.
   Zenith Z-100 - DSDD 48 tpi 5.25" format

   B0:B
   288K free on drive B:
   Press any key to continue...
```

Hah! I messed up when I entered a drive letter! I reformatted and tried again. This time I entered the DOS file name as **E:\DATETIME\*.*** and instead of entering a drive letter when the computer responded with "*Enter CP/M destination name*", I just pressed **ENTER**:

After the standard heading display, I got what I wanted:

```
   Copying E:\DATETIME\DIR.COM to B0:DIR.COM
   Copying E:\DATETIME\DATE.COM to B0:DATE.COM
   Copying E:\DATETIME\SHOW.COM to B0:SHOW.COM

   274K remaining on dive B:
   Press any key to continue...
```

Finally! Checking the directory again showed all was well:

```
   (CDIR - standard heading)

   Zenith Z-100 - DSDD 48 tpi 5.25" format

   B0:DATE.COM   B0:DIR.COM   B0:SHOW.COM

   274K free on drive B:
   Press any key to continue...
```

Now, to erase these files, I pressed **8** and the computer responded with:

```
   ERASE (DELETE) CP/M FILE(S)

   Enter CP/M file name. If you specify a
   diskette drive (user number is optional),
   it will override the selected CP/M drive.
   You may include 'wild card' characters
   (* and ?) in the name.
```

So, we know now **NOT** to enter a drive letter. I just entered ***.*** and held my breath that it wouldn't delete my hard drive.

```
   CERA Ver. 1.44-Oct 31 1996, Copyright
   1996, Sydex. All rights reserved.
   THIS IS AN UNREGISTERED COPY --
   SEE DOCUMENTATION FOR DETAILS.
   Zenith Z-100 - DSDD 48 tpi 5.25" format

   Deleted DIR.COM
   Deleted DATE.COM
   Deleted SHOW.COM

   304K free on drive B:
   Press any key to continue...
```

Great! Now, let's try from my other floppy drive, drive A:, a 3.5" floppy containing those same three files from s100computers.com. We were back to our CMenu screen, so I entered **4**, and the computer responded with:

```
   COPY DOS FILES TO CP/M

   Enter DOS file name (e.g., C:\MYDIR\*.TXT)
```

I entered **A:\DATETIME\*.*** and the program responded with:

```
   Enter CP/M destination name... (as before)
```

Do **NOT** enter **B:** as a destination drive - we saw the result above. Just press **ENTER** and the computer responded with:

```
   (DTOC - standard heading)

   Zenith Z-100 - DSDD 48 tpi 5.25" format

   Insert DOS diskette in drive B:
   Press any key when ready
```

Well, I was stumped. What was it asking? I wanted the files from A:, maybe it is just a typo? So, I just pressed **ENTER**.

The program responded with:

```
   Insert DOS diskette in drive B:
   Press any key when ready...
```

Now what? I just pressed **ENTER**,

```
   Insert CP/M diskette in drive B:
   Press any key when ready
   Copying A:\DATETIME\DIR.COM to B0:DIR.COM

   Insert DOS diskette in drive B:
```

```
  Press any key when ready...
```

Pressed **ENTER** again,

```
  Insert CP/M diskette in drive B:
  Press any key when ready
  Copying A:\DATETIME\DATE.COM to
  B0:DATE.COM

  Insert DOS diskette in drive B:
  Press any key when ready...
```

Pressed **ENTER** again,

```
  Insert CP/M diskette in drive B:
  Press any key when ready
  Copying A:\DATETIME\SHOW.COM to
  B0:SHOW.COM

  Insert DOS diskette in drive B:
  Press any key when ready...
```

And yet **ENTER** once more. The program finished with:

```
  274K remaining on drive B:
  Press any key to continue...
```

Success, but obviously the program just didn't like copying from the A: drive (a bug?). Checking the directory again showed all was well:

```
  (CDIR - standard heading)
  Zenith Z-100 - DSDD 48 tpi 5.25" format

  B0:DATE.COM    B0:DIR.COM    B0:SHOW.COM

  274K free on drive B:
  Press any key to continue...
```

Thinking the problem may have been having the desired files in a sub-directory on drive A, I copied another file, SETDEF.COM, onto drive A: in the root directory using Windows, then tried the above again. Nope, the program still went through the same 'Insert disk' procedure.

Now it was time to check the disk on my Z-100 and try copying the files to the Z-100's hard drive. I'll be back in a minute...

Nope, it didn't work! On **A>DIR B:**, I got:

```
  Format Error
  Unable to Determine Disk Format And
  Complete Disk Select Operation.

  CP/M Error On B: Invalid Drive
  BDOS Function = 17 File = ????????.???
  A>
```

I tried **A:STAT B:**, and got the same message down to:

```
  BDOS Function = 14
```

So I reformatted the disk on the Z-100 using the DSDD 8*512 Format (320K) option and brought it back to my PC.

This 22Disk could read just fine. I copied the files from it to hard drive D: with 22DISK's function #3.

```
  COPY CP/M FILES TO DOS

  Enter CP/M file name... (as before)
```

I entered ***.*** and the program responded with:

```
  Enter DOS destination with drive and path
  (E:\22DISK assumed) -_
```

I entered **E:\CPM3filz** and the copying commenced:

```
  (CTOD - standard heading)
  Zenith Z-100 - DSDD 48 tpi 5.25" format

  Copying B0:filename.ext to
  E:\CPM3FILZ\filename.ext
  (... more)
  (... more)
  13 file(s) copied
  Copy complete

  Press any key to continue -_
```

Now, to erase these files, I pressed **8**, and the computer responded as before. After typing *.*, the files were deleted.

22Disk comes with a utility that can automatically sense the floppy drive configuration of your PC. Since I had difficulty with drive A: and drive B: being sensed properly during the copy evolution, I double-clicked on VIEWCONF from within Windows and a DOS Window opened up to display the following:

```
  Sydex Diskette Configuration Viewer
       Copyright 1994, Sydex, Inc.

  The configuration routine did not find a
  DISKETTE.CFG* file and will determine the
  configuration automatically.
```

Then within a graphics box the program displayed:

```
DRIVE TYPE BIOS PORT UNIT DMA  INT  SEEK  SPECIAL  P
 A:   1.44M  0   3F0h   0   2    6    6            0
 B:    360K  1   3F0h   1   2    6    6            0

  SPECIAL flags:
  CH - Change Line,CC - CompatiCard I/II,
  DU - Dual Speed Motor,2X - Double Speed Drive
  "P" column indicates logical data path number
  assignment
```

Note *: DISKETTE.CFG can be written in any editor to configure the drives manually. As this looked fine, I left it alone.

Finally, I loaded CP/M-85 into my Z-100 and tried reading the 22Disk formatted disk again. CP/M-85 read the disk just fine. I formatted another disk using CP/M-85's FORMAT program and took it back to 22DISK to copy the desired files to it. Again, 22Disk read and wrote to the disk just fine. Returning to CP/M Plus, the files on this new disk were also just fine.

So, in summary, CP/M-85 reads the 22Disk formatted disks just fine, but CP/M Plus does NOT. The problem must be in CP/M Plus - another problem to address in the future. For now, just format any disks that you want to transfer data with from CP/M Plus and the process will work fine.

Date Patch For CP/M 3's DATE.COM

As I mentioned earlier, DATE.COM and a few other programs using the date have a Y2K problem, in that they cannot handle dates after December 31, 1999. I searched the internet and found a PATCH data file, in a format I couldn't follow, but I managed to figure out what I needed in order to use SID to change the correct bytes in DATE.COM.

Briefly, we will enter the following new bytes beginning at each address listed:

```
Addr: New bytes:
0110  FE 4E D2 17 01 C6 64 32 98 0B C9
0120  79 06 13 FE 64 DA 2B 01 04 DE 64
      F5 48 CD 72 05 F1 4F C3 72 05
0750  00
0754  CD 10 01
0963  20 01
0A1F  16
```

Unassembling the file following the patch, the code becomes:

```
Addr: Code:
010D  NOP  ;We are in Page Zero
010E  NOP  ;The area below the TPA
010F  NOP  ;(Temporary Program Area)
0110  CPI  4E  ;4Eh= 78d= 1978 (min year)
0112  JNC  0117
0115  ADI  64     ;Add 64h or 100d years
0117  STA  0B98   ;Save at addr 0B98
011A  RET

0120  MOV  A,C
0121  MVI  B,13   ;13h= 19d
0123  CPI  64     ;64h= 100d
0125  JC   012B
0128  INR  B
0129  SBI  64     ;Sub 100d
012B  PUSH PSW
012C  MOV  C,B
012D  CALL 0572
0130  POP  PSW
0131  MOV  C,A
0132  JMP  0572

074F  MVI  C,0    ;Was 4Eh, the min. year
                  ;that could be input

0754  CALL 0110   ;Call first new routine

0962  CALL 0120   ;Call second new routine

0A1E  MVI  A,16   ;Was 14h
```

To perform the patch to DATE.COM, we use the SID utility, which is very similar to DEBUG under DOS. The command would be:

```
SID DATE.COM     for the default drive, or
SID B:DATE.COM   if using a B: drive
```

This loads the utility and the program DATE.COM into memory:

```
CP/M 3 SID - Version 3.0
NEXT MSZE  PC  END
0C00 0C00 0100 D9FF
#
```

At the # sign, we can use any of a variety of commands. For example, to just dump the first page of our DATE program in memory we use the '**D**'ump command. Remember, our .COM program begins at 0100, so enter the command **D0100** at the # sign. The computer would respond with something that appears like:

```
(addr): .. (16 data bytes) .. (16 ASCII)
0100: 31 5F 0B C3 95 02 00 00 00 .etc.
      followed by: 1_.............
0110: all zeros (This is where our    )
0120: all zeros ( new routines will go)
0130: all zeros
0140: 43 50 2F 4D 20 56 65 72 73 ..etc.
      followed by: CP/M Version 3.0
0150: etc. to 01B0:
```

Pressing '**D**'ump again, would continue with the next 12 rows of 16 bytes.

To change the data at a certain address we use the '**S**'et command. For our patch, we want to begin at address 0100, so we enter the command:

#S0110

The computer jumps to that address and displays the address and the byte value located there, such as:

```
0110 00
```

and waits for you to input a new value, or, if you want to leave it unchanged, press a period {.} and {RETURN}. A period exits the set command and returns us to the prompt # sign.

If you enter a value and press return the program goes to the next address, displays it, and waits for your input. The process continues until you press a period {.}.

So for our first patch, the display would look like this:

```
0110 00 FE{CR}
0111 00 4E{CR}
0112 00 D2{CR}
0113 00 17{CR}
0114 00 01{CR}
0115 00 C6{CR}
0116 00 64{CR}
0117 00 32{CR}
0118 00 98{CR}
0119 00 0B{CR}
011A 00 C9{CR}
011B 00 .{CR}
```

Use the '**S**'et command in a similar manner to perform the remaining patches as listed above.

Once done, you can confirm the patches by using the '**D**'ump command again, or if you wish to see the actual code, try the '**L**'ist command. The '**L**'ist command uses the syntax: **L**{s}{,f} where 's' is the start address and 'f' is the finish address. So to see the new code, just enter:

```
#L0110,0130{CR}
```

You should see the list of statements that I presented above. Code is listed up to 12 lines at a time; just press {RETURN}, if needed, to continue. Now try this with the other sections of changed code.

When you are finished, we need to save our newly patched program. We use the '**W**'rite command, including the file name, as:

```
#WDATE.COM     on the default drive, or
#WB:DATE.COM   on the B: drive
```

Now try running **DATE SET** or **B:DATE SET**, as appropriate, at the system prompt.

Closing

I hope you enjoyed our trip into the distant past and our jaunt into the world of CP/M. If you are so inclined, I hope you will give the above procedures a try. In the next issue I hope to continuing learning a little more by digging a bit deeper into CP/M. Specifically, I want to disassemble DATE.COM, fix the date, and make the program more useful by getting the program to display the date/time on the screen. I hope you will join me.

Again, I want to thank all of you who renewed your subscriptions. We even have a few new ones. I hope I'm meeting your expectations.

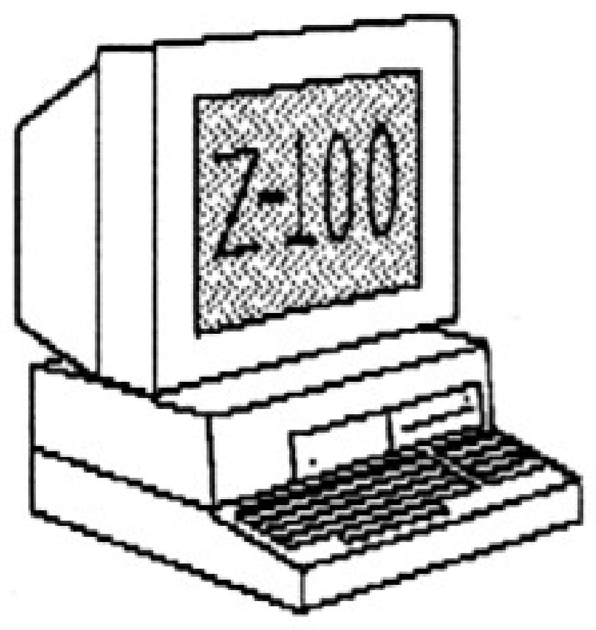
Remember, I'm no longer on a set publishing schedule, so don't get too excited if you don't get an issue for an extended period, especially this next one. I might be biting off more than I can chew at the moment.

If you wish to help me, I would appreciate it. I'd be happy to set you up and get you started. Also, please feel free to contact me with suggestions on other changes for CP/M while we are here.

Cheers!!!

'Til next time,
happy computing!

[illegible]



Z-100 LIFELINE
Supporting the H/Z-100 Community Since 1989

Steven W. Vagts
211 Sean Way
Hendersonville, NC 28792
(828) 685-8924
Email: z100lifeline@swvagts.com

Don't forget our Z-100 LifeLine Web Site:

https://z100lifeline.swvagts.com

Z-100 Parts & Service
DOS v3 Software & Documentation
MTR-ROM v4 & Z-DOS v4 Software
Z-207 w/High Density Drive mods
Z-205 Mods and RAMDisk Software
Past "Z-100 LifeLines" on CD-ROM
Z-100LL Software Library on CD-ROM
Z-100 LifeLine Archive Library
New IDE Controller Card
"We've got most everything!"

```
;           All-Base Number Conversion Program for CP/M
; Created from the Heathkit Assembly Language Programming Course
; Modified by Steven W. Vagts, Editor, Z-100 LifeLine, 8/10/12

;BDOS Calling Conventions:
;CP/M 3 and this program use a standard convention for BDOS function calls.
;Entry:      [C] contains the BDOS function number
;            [DE] contains a byte or word value or an information address.
;Retns:      [A] contains single-byte values
;            [HL] contains double-byte values; 0FFFFh on error.

BDOS         equ   0005h ;BDOS entry point for BDOS calls

;FunctionName      Value:      ;Func# (dec) & Description:
Boot         equ   0000h       ; 0 - System Reboot or Reset
ConIn        equ   0001h       ; 1 - Console Input
Conout       equ   0002h       ; 2 - Console Output
PString      equ   0009h       ; 9 - Print String

;Other useful equates are:
CR           equ   0Dh         ;Carriage Return
LF           equ   0Ah         ;Line Feed

;Following all these equates, we can begin the code:
      ORG   0100        ; Beginning of TPA

Start:
      LXI   SP,STACK    ; Create Local Stack at 0B5F
      JMP   Begin       ; Jump over any data area

Msg1:
      DB  1Bh,45h,CR,LF,'                '
      DB  1Bh,70h,'  All-Base Number Conversion Program  ',1Bh,71h,CR,LF
      DB  CR,LF,'Created by Steven W. Vagts, Z-100 LifeLine, 7/30/2012'
      DB  CR,LF,LF,'This program will convert:',CR,LF
      DB  '    ',1Bh,70h,'B',1Bh,71h,'inary (limit to 16 bits)',CR,LF
      DB  '    ',1Bh,70h,'H',1Bh,71h,'ex (max FFFFh)',CR,LF
      DB  '    ',1Bh,70h,'O',1Bh,71h,'ctal (max 177777)',CR,LF
      DB  '    ',1Bh,70h,'D',1Bh,71h,'ecimal (max 65535), and',CR,LF
      DB  '    ',1Bh,70h,'A',1Bh,71h,'SCII characters.',CR,LF
      DB  'Use ',1Bh,70h,'X',1Bh,71h,' to exit.',CR,LF,LF
      DB  'The program will display the value given in each of the other '
      DB  'number bases,',CR,LF,'and will display the BCD value, and '
      DB  'the graphics char, if applicable.$'
;     Note: Wrap around occurs if the limits above are exceeded, which
;           can cause some confusion. For example: 65536d becomes 00000d.
;           Also, exceeding the number of digits or characters will
;           cause truncation from the left. For example, ASCII 'ABC'
;           will be processed as 'BC'; Likewise, 89ABCh will be '9ABC'.
Msg2:
      DB  CR,LF,LF,'What Base is your entry < A, B, D, H, O or X >? $'
Msg3:
      DB  CR,LF,'Please enter your number or ASCII character:  $'
MsgCvt1:
      DB  CR,LF,'  Hex   Octal  Decimal  Binary_Coded_Decimal',CR,LF,'  $'
MsgCvt2:
      DB  CR,LF,'  Binary                ASCII  Graphic',CR,LF,'  $'
MsgErr:
      DB  CR,LF,'Sorry, I don''t understand!$'
MsgErr1:
      DB  CR,LF,'Input character wrong for Base specified.$'
BCDcnt:
      DB  0             ;Number of char in MsgBCD
MsgBCD:                 ;BCD buffer
      DB  0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,CR,LF,'$'
MsgASC:                 ;Create ASCII string buffer
      DB  20h,20h,20h,20h,00h,20h,00h,20h,20h,20h,20h  ;ASCII string
```

```
MsgGFx:                 ;Create Graphic string buffer
      DB  1Bh,46h,00h,20h,00h,1Bh,47h      ;ESC-F, grafx chars, ESC-G
      DB  CR,LF,'$'

;Now we come to the real coded statements:
Begin:
      LXI   D,Msg1            ;Display opening Title and message.
      MVI   C,PString         ;BDOS func#9, Pring String
      CALL  BDOS
Input:
      LXI   D,Msg2            ;Request Number Base
      MVI   C,PString         ;BDOS func#9, Print String
      CALL  BDOS
GetBase:
      MVI   C,ConIn     ;BDOS func#1, Console Input
      CALL  BDOS        ;A = char
      ANI   07Fh        ;Strip parity
      ANI   05Fh        ;Convert to Upper Case
      CPI   'A'         ;Is it ASCII?
      JZ    AScin       ;Do ASCII
      CPI   'B'         ;Is it Binary?
      JZ    BINin       ;Do Binary
      CPI   'D'         ;Is it Decimal?
      JZ    DECin       ;Do Decimal
      CPI   'H'         ;Is it Hexadecimal?
      JZ    HEXin       ;Do Hex
      CPI   'O'         ;Is it Octal?
      JZ    OCTin       ;Do Octal
      CPI   'X'         ;All done
      JZ    Quit
      LXI   D,MsgErr    ;Error
      CALL  PMsg
      JMP   Input       ;Return for more

AScin:
      CALL  ConInMsg
NxtAChr:
      CALL  RdCon       ;A = ASCII char
      CPI   00Dh        ;{CR}?
      JZ    Cnvert      ;Done, go convert to all BASEs
      MOV   D,E         ;Last Char moves to D
      MOV   E,A         ;New Char goes in E
      JMP   NxtAChr     ;Loop for next char

BINin:
      CALL  ConInMsg
NxtBChr:
      CALL  RdCon       ;A = number char
      CPI   00Dh        ;{CR}?
      JZ    Cnvert      ;Done, go convert to all BASEs
      SUI   030h        ;A = A-30h
      JC    Error       ;No good if Chr<30h
      CPI   002h        ;Higher than "1"?
      JNC   Error       ;No good for Decimal #
      LXI   H,0000h     ;Clear HL for multiplication
      DAD   D           ;HL = HL+DE (Times 1)
      DAD   H           ;HL = HL+HL (Times 2)
      MOV   E,A         ;Place char in E
      MVI   D,00h       ;Clear D register
      DAD   D           ;HL = HL+DE, Add unit value to DE
      XCHG              ;HL<>DE, new total to DE
      JMP   NxtBChr     ;Loop for next char
```

```
DECin:
      CALL   ConInMsg
NxtDChr:
      CALL   RdCon        ;A = number char
      CPI    00Dh         ;{CR}?
      JZ     Cnvert       ;Done, go convert to all BASEs
      SUI    030h         ;A = A-30h
      JC     Error        ;No good if Chr<30h
      CPI    00Ah         ;Higher than "9"?
      JNC    Error        ;No good for Decimal #
      LXI    H,0000h      ;Clear HL for multiplication
      DAD    D            ;HL = HL+DE (Times 1)
      DAD    H            ;HL = HL+HL (Times 2)
      DAD    H            ;HL = HL+HL (Times 4)
      DAD    D            ;HL = HL+DE (Times 5)
      DAD    H            ;HL = HL+HL (Times 10)
      MOV    E,A          ;Place char in E
      MVI    D,00h        ;Clear D register
      DAD    D            ;HL = HL+DE, Add unit value to DE
      XCHG                ;HL<>DE, new total to DE
      JMP    NxtDChr      ;Loop for next char

HEXin:
      CALL   ConInMsg
NxtHChr:
      CALL   RdCon        ;A = number char
      CPI    00Dh         ;{CR}?
      JZ     Cnvert       ;Done, go convert to all BASEs
      SUI    030h         ;A = A-30h
      JC     Error        ;No good if Chr<30h
      CPI    00Ah         ;Higher than "9"?
      JC     HexOk        ;It's OK for Hex#
      SUI    007h         ;Subtract 07h more for letters
      CPI    00Ah         ;<0Ah?
      JC     Error        ;No good if so
      CPI    010h         ;Higher than "F"?
      JNC    Error        ;No good if so
HexOk:
      LXI    H,0000h      ;Clear HL for multiplication
      DAD    D            ;HL = HL+DE (Times 1)
      DAD    H            ;HL = HL+HL (Times 2)
      DAD    H            ;HL = HL+HL (Times 4)
      DAD    H            ;HL = HL+HL (Times 8)
      DAD    H            ;HL = HL+HL (Times 16)
      MOV    E,A          ;Place char in E
      MVI    D,00h        ;Clear D register
      DAD    D            ;HL = HL+DE, Add unit value to DE
      XCHG                ;HL<>DE, new total to DE
      JMP    NxtHChr      ;Loop for next char

OCTin:
      CALL   ConInMsg
NxtOChr:
      CALL   RdCon        ;A = number char
      CPI    00Dh         ;{CR}?
      JZ     Cnvert       ;Done, go convert to all BASEs
      SUI    030h         ;A = A-30h
      JC     Error        ;No good if Chr<30h
      CPI    008h         ;Higher than "7"?
      JNC    Error        ;No good for Octal #
      LXI    H,0000h      ;Clear HL for multiplication
      DAD    D            ;HL = HL+DE (Times 1)
      DAD    H            ;HL = HL+HL (Times 2)
      DAD    H            ;HL = HL+HL (Times 4)
      DAD    H            ;HL = HL+HL (Times 8)
```

```
        MOV   E,A           ;Place char in E
        MVI   D,00h         ;Clear D register
        DAD   D             ;HL = HL+DE, Add unit value to DE
        XCHG                ;HL<>DE, new total to DE
        JMP   NxtOChr       ;Loop for next char

Cnvert:
        PUSH  D             ;Save value in DE
        LXI   D,MsgCvt1     ;Set DE to Msg addr
        CALL  PMsg
        POP   D             ;Get DE value back

Hout:         ;Convert unknown number to Hexadecimal
;Entry:       DE = the binary unknown value to convert
;To convert our binary number to hexadecimal, we must take four bits at
;a time, add 030h to make it ASCII, and check to see if the result is
;"higher" than the ASCII code for "9". If it is, just add 070h more
;and display the character.
;Note: We must rotate the four most significant bits of a data byte
;into the four least significant positions by rotating the accumulator
;to make it easy to convert to ASCII and display the character.

        MOV   A,D           ;A = High byte of unknown value
        CALL  DoHex         ;Convert to HEX and display
        MOV   A,E           ;A = Low byte of unknown value
        CALL  DoHex         ;Convert to HEX and display
        CALL  Space2        ;Display two spaces

Oout:         ;Convert unknown number to Octal
;Entry:       DE = the binary unknown value to convert
;To convert our binary number to octal, the process is similar to the
;steps used for conversion to HEX, except the first octal character is
;made up of only the leftmost one bit of the data byte in register D.
;Then there are two pretty easy characters. The hard part is the third
;character, which has one bit from the data byte in D and two bits from
;the data byte in E.

        MOV   A,D           ;A = High byte of unknown value
        RAL                 ;Rotate leftmost bit
        RAL                 ;  into rightmost position
        PUSH  PSW           ;Save new byte to Stack
        ANI   001h          ;Zero all but right bit
        CALL  OCTout        ;Make ASCII and display
        POP   PSW           ;Get modified byte back
        CALL  ROTes         ;Do next three bits 2 times
        MOV   A,E           ;A = Low byte of unknown value
        CALL  ROTes         ;Do leftover of high and 5 of low
        CALL  ROTer         ;Do last three bits
        CALL  Space2        ;Display two spaces

Dout:         ;Convert unknown number to Decimal
;Entry:       DE = the binary unknown value to convert
;To convert our binary number to decimal, we must subtract the decimal
;positional value from the unknown binary value and keep count of the
;number of times. If the subtraction generates a carry (borrow), we
;subtracted one too many times.
;As we find the number of successful subtractions for each position, we
;can display the number on the screen. Repeat for each decimal position.

        XCHG                ;HL<>DE ;Swap unknown into HL
        PUSH  H             ;Save unknown onto Stack
        LXI   B,10000d      ;BC = 10000d
        CALL  Dsub          ;Call the subtraction routine & display
        LXI   B,1000d       ;Repeat for BC = 1000d position
        CALL  Dsub
```

```
      LXI    B,100d       ;Repeat for BC = 100d position
      CALL   Dsub
      LXI    B,10d        ;Repeat for BC = 10d position
      CALL   Dsub
      MOV    A,L          ;What's left is units only
      PUSH   PSW          ;Save our decimal digit
      ADI    030h         ;Make it ASCII
      CALL   WrtCon
      CALL   Space2
      CALL   Space1       ;Need three spaces
      POP    PSW          ;Get it back
      CALL   BinBCD       ;Convert to BCD and save
      LXI    D,MsgBCD     ;DE = MsgBCD addr
      CALL   PMsg
                          ;Now clean up
      LDA    BCDcnt       ;A = BCD char count
      MOV    C,A          ;C = BCD count
      MVI    A,00h
      STA    BCDcnt       ;Zero BCD count
      LXI    D,MsgBCD     ;DE = MsgBCD base addr
BCD0:
      STAX   D            ;Zero MsgBCD char
      INX    D            ;Next addr
      DCR    C
      JNZ    BCD0         ;Zero next addr
Cnvert2:
      LXI    D,MsgCvt2    ;Set DE to Msg addr
      CALL   PMsg
      POP    D            ;Get Original unk value back

Bout:        ;Convert unkwn value in DE (already in binary for display)

      MOV    A,D          ;A = High byte of unknown value
      CALL   BINout       ;Display as binary
      CALL   Space1       ;Display a space
      MOV    A,E          ;A = Low byte
      CALL   BINout       ;Display as binary
      CALL   Space2       ;Display 2 spaces

Aout:
;To convert the unknown value in DE to ASCII, all you have to do is
;output each byte. However, the bytes must be tested to see if they are
;printable ASCII characters. Some of the control codes could cause
;strange things to happen if they were output to the display.

      PUSH   D            ;DE has unknown value
      MOV    A,D          ;A = High byte of unknown
      LXI    D,MsgASC+4   ;Set ASCII string addr
      PUSH   PSW          ;Save char
      CALL   Atest        ;Test for legal and store
      POP    PSW          ;Get char back
      LXI    D,MsgGFx+2   ;Set ASCII Graphic addr
      CALL   Gtest        ;Test for legal and store
      POP    D            ;Get unknown value again
      MOV    A,E          ;A = Low byte
      LXI    D,MsgASC+6   ;Set ASCII string addr
      PUSH   PSW          ;Save char
      CALL   Atest        ;Test for legal and store
      POP    PSW          ;Get char back
      LXI    D,MsgGFx+4   ;Set ASCII Graphic addr
      CALL   Gtest        ;Test for legal and store
Aout1:                    ;Display the MsgASC String
      LXI    D,MsgASC     ;Set base addr of string
      CALL   PMsg
      JMP    Input        ;Go for another round
```

```
DoHex:
      PUSH  PSW         ;Save byte on Stack
      RRC               ;Rotate
      RRC               ;  left 4 bits
      RRC               ;  into least
      RRC               ;  four bits
      CALL  HEXout      ;Make ASCII and display
      POP   PSW         ;Get Byte back
HexOut:
      ANI   00Fh        ;Keep only right four bits
      ADI   030h        ;Add ASCII offset
      CPI   ':'         ;Is it a number?
      JC    WrtCon      ;Display, if so
      ADI   007h        ;Make it a letter
      JMP   WrtCon      ; and display

ROTes:
      CALL  ROTer       ;Do 3 bits and fall into 3 more
ROTer:
      RAL               ;Rotate carry plus leftmost
      RAL               ;  two bits of data byte into
      RAL               ;  three rightmost bits
OCTout:
      PUSH  PSW         ;Save modified byte on Stack
      ANI   007h        ;Zero all but right three bits
      ADI   030h        ;Add in ASCII offset
      CALL  WrtCon      ;Display Char
      POP   PSW         ;Get modified byte back
      RET               ;Return to main program

Dsub:
      MVI   D,0FFh      ;Set our counter to -1 to start
Dsub1:
      MOV   A,L         ;A = Low byte of unknown
      SUB   C           ;Subtract low byte of divisor
      MOV   L,A         ;Difference back to L; CY=1, if borrow
      MOV   A,H         ;A = High byte of unknown
      SBB   B           ;A = A-B-CY; If carry, must subtract 1
      MOV   H,A         ;Difference back to H
      INR   D           ;Count one subtraction
      JNC   Dsub1       ;Subtract again if no borrow from SSB
                        ;However, if borrow occurred, we went one
      DAD   B           ; too far and have to add divisor back
      MOV   A,D         ;A = Count
      PUSH  PSW         ;Save our decimal digit, we're not done
      ADI   30h         ;Make it ASCII
      CALL  WrtCon      ;And display on CRT
      POP   PSW         ;Get it back
BinBCD:                 ;Fall into converting to BCD; A=Dec char
      RAL               ;Only want the last 4 bits, so skip 4
      RAL
      RAL
      RAL
      MVI   B,04h       ;4 bits to do
BCDloop:
      RAL               ;Rotate a bit into Carry
      PUSH  PSW         ;Save the modified byte
      MVI   A,'1'       ;Assume bit is logic 1
      JC    BCDok       ;Store it if so
      MVI   A,'0'       ;No, it was 0
BCDok:                  ;A has binary bit
      CALL  BCD1        ;Save A in BCD buffer
      POP   PSW
      DCR   B
      JNZ   BCDloop     ;Do another until 4 done
```

```
        MVI     A,20h           ;We want a space every 4 bits
        CALL    BCD1            ;Save space in BCD buffer
        RET

BCD1:
        PUSH    B               ;Save registers
        PUSH    D
        PUSH    H
        PUSH    PSW             ;Save char
        LDA     BCDcnt          ;A = BCD char count
        MOV     C,A             ;Move count to C
        MVI     B,00h
        LXI     H,MsgBCD        ;HL = BCD buffer addr
        DAD     B               ;HL = HL+BC
        XCHG                    ;HL<>DE
        INR     C
        MOV     A,C
        STA     BCDcnt          ;Store count
        POP     PSW             ;Get back binary bit
        STAX    D               ;Store binary bit
        POP     H
        POP     D
        POP     B
        RET

BINout:
        MVI     B,008h          ;8 bits to do
Bloop:
        RAL                     ;Rotate a bit into Carry
        PUSH    PSW             ;Save the modified byte
        MVI     A,'1'           ;Assume bit is logic 1
        JC      BINok           ;Display it if so
        MVI     A,'0'           ;No, it was 0
BINok:
        CALL    WrtCon          ;Display whatever
        POP     PSW             ;Get modified byte back
        DCR     B
        JNZ     Bloop           ;Do another until 8 done
        RET

Atest:
        CPI     020h            ;Lower than a space?
        JC      DoSpc           ;Place space in ASCII buffer
        CPI     07Fh            ;Lower than 7Fh?
        JNC     DoSpc           ; No, place space in ASCII string
        CALL    DoChr           ;Store legal char
        RET

Gtest:                          ;Test if graphics char
        CPI     05Eh            ;Lower than a '^'?
        JC      DoSpc           ;Store space
        CPI     07Fh            ;Lower than 7Fh?
        JC      DoChr           ;Store char in ASCII string
DoSpc:
        MVI     A,20h           ;Store space in Grafx string
DoChr:
        STAX    D               ;Store A=char in DE
        RET

CRLF:
        MVI     A,0Dh           ;CR
        CALL    WrtCon
        MVI     A,0Ah           ;LF
        JMP     WrtCon
```

```
Space2:
       MVI    A,' '         ;A = space
       CALL   WrtCon        ;Display it
Space1:
       MVI    A,' '         ;A = space
       JMP    WrtCon        ;Display a second and RET

Error:
       LXI    D,MsgErr1     ;Error Message
       CALL   PMsg
       JMP    Input

Quit:
       MVI    C,Boot        ;BDOS func#0 - System Reset
       CALL   BDOS
ConInMsg:
       LXI    D,Msg3        ;Request Number or ASCII char
       CALL   PMsg
       LXI    D,0000h       ;Zero DE registers
       RET

RdCon:
       PUSH   B
       PUSH   D
       PUSH   H
       MVI    C,ConIn       ;BDOS Func#1, Read Console Char
       CALL   BDOS          ;A = char input
       POP    H
       POP    D
       POP    B
       RET

WrtCon:
       PUSH   B             ;Save Registers
       PUSH   D
       PUSH   H
       MOV    E,A           ;E = char to output
       MVI    C,ConOut      ;BDOS Func#2, Write to screen
       CALL   BDOS
       POP    H
       POP    D
       POP    B
       RET

PMsg:
       PUSH   B
       PUSH   H
       PUSH   PSW           ;Save char to Stack
       MVI    C,PString     ;BDOS func#9 - Print String
       CALL   BDOS
       POP    PSW           ;Restore char
       POP    H
       POP    B
       RET

;End of code area
       DS   014h            ;Reserve space for Stack
Stack  DS   001h            ;Top of Stack is here

       END    Start         ;End of program
```