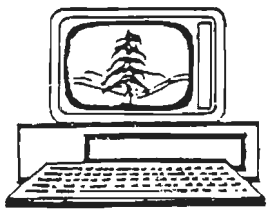

The Stanford/Palo Alto Users Group for the IBM PC

PO Box 8292 Stanford, CA 94305

NEWSLETTER VOLUME 1 NUMBER 4 JULY 1983

<<<< THERE HAS BEEN A ROOM CHANGE FOR THIS MEETING >>>>
(again)

Please note that we are meeting at the Graduate School of Business on Stanford Campus.

****
* Next meeting: Wednesday, July 27, 1983 *
* 7:00pm *
* Rm 74, Graduate School of *
* Business, Stanford University *
****

AGENDA FOR THE MEETING --

7:00 General Club Business
Report on the Wednesday Before the Monday Meeting
New Business

7:40 Commencement of Novice Night--Short Discourses and long Q & A periods regarding hardware, software, and your general karma with your IBM PC.

8:45 Random Access

9:00 Conclusion of the Official Club Meeting

Reminder--Don't forget about the Monday Before the Wednesday Meeting at Talbott's on California Ave, across the street from the Winery. Time is at 8:30 pm, goes on until whenever (usually about an hour and a half or so). All are invited to this meeting where great plans are made and policy discussed. We will be discussing how much to raise club membership fees, so bring your calculators and your Compaq's.

PERIODICAL REVIEW

Just in time for novice night, PC Magazine offers half a dozen articles on training, including reviews of top-selling training programs. Also an amusing story of a first-time buyer's experiences which begins on page 379.

The focus at PC World this month is communication packages. Remember while reading reviews of the various communication packages available that PC Talk III is "Freeware" and resides in our club library.

PC Age is probably this month's most useful magazine. Included are articles on Do-it-yourself maintenance, diagnostics, dot-matrix printer repair and care, and a couple of articles/reviews for DOS 2.0. This is all in Volume 2.6.

Want some inside scoop on Visi On, Microsoft's multi-operatingoperating system? Look to page 166 in the July issue of Byte.

Interested in Machine-Language programs? That would also be in Byte, on page 398.

Assembly-language is covered in the first of three parts in the current issue of PC World. This would be somewhat of a tutorial, and might be somewhat of a worthwhile investment as IBM (and all their devoted followers) will be using Intel's chips for sometime to come (IBM owns a few shares of Intel). A quick review of Intel chips on the horizon starts on page 40. I would like to see this new text co-processor, the 82730, in action. It allows your screen to show italics, bold, super and subscripts, etc.

Also in PC World this month is another convert's report on his adoption of the language FORTH. Some dare call it a religion. At any rate, read all about it in further pages of this newsletter, as Pardner Wynn gives a quick introduction to this very adaptive language.

*** Some documentation for the MVPFORTH ***
*** package on the SPACL F-00 and F-01 ***
*** club library disks ***

Pardner Wynn 7-7-83
P.O. Box 3006
Stanford, Ca 94305

NOTE:
This file is in no way an introduction to Forth. It is mainly an attempt to very briefly describe the basic structure of the two MVPFORTH disks in the Stanford/Palo Alto PC User's Club library.

For an EXCELLENT treatment of Forth, try Leo Brodie's STARTING FORTH, available everywhere. Mountain View Press, P.O.Box 4656, Mountain View, Ca 94040, carries a complete line of Forth reference books, disks, and the like — including STARTING FORTH.

0) There are 4 files on disk F-00:

COMMAND.COM	
MVPFORTH.ASM	— MVPFORTH source code
MVPFORTH.EXE	— executable MVPFORTH
SPACLF00.DOC	— club librarian's .DOC file

Refer to section (2) below for a how-to-use.

1) SOME GENERAL COMMENTS ON HOW FORTH TREATS DISK FILES

The second disk, F-01, is comprised entirely of Forth 'screens'. Most versions of Forth do not use standard disk files for storage; they use their own disk-handling routines to control access to the disk. Therefore, to access a Forth file, you need to be running the Forth program, and use the Forth routines provided to edit, compile, and copy programs.

This is how Forth treats disks, and disk files. Forth treats a disk like a very large sequential set of files, called SCREENs, each of which has 16 lines of 64 characters each. That's all a Forth screen (or, file) is: an area on the disk with room for 16x64 = 1024 (1K) characters. In fact, Forth doesn't even give names to each screen, just a number. And that number actually tells Forth WHERE on the disk that file is.

Example of SCREEN —> DISK mapping

As an example, suppose you have a floppy disk with room for 160K. To Forth, that disk is a collection of 160 screens, each 1K long. And screen #0 is defined to be the first 1K block of space on the disk; screen #1 is the second 1K block , and so on, until screen #159, which is the 160th (and last) 1K block.

If you have two floppy drives, the screens on drive B: take over from where drive A: leave off. That is, the 0th screen on drive B: is actually screen #160 (159 + 1) to Forth for a system with two 160K disks.

Perhaps you have a short routine you've written in Forth that is called GET-A-NUMBER, which asks the user for a number, then stores it in a variable. If you used the Forth editor to put that Forth program on screen #50 (drive A:), then that program is on your disk at the 50th (if you count by starting with zero) physical 1K b on your disk. To look at that program, you'd type 50 LIST, which tells Forth to go to screen #50, read all 1K characters into memory (even if blanks), then type them out on your screen.

How to keep track of what's where

turns out to be very fast, and quite powerful, but it does have a drawback: YOU HAVE TO KEEP TRACK OF THE SCREEN NUMBERS THAT YOUR FORTH PROGRAMS ARE STORED ON. Two 'conventions' make this easier than it sounds.

First, most Forth programmers use line #0 of each screen to tell what is on that screen. Forth's INDEX command allows you to see line #0 for a range of screens, so typing 0 10 INDEX types out line #0 of screens #0 thru #10, which should show you what is on those screens.

Second, most Forth programmers also include a DIRECTORY SCREEN, which is simply a screen with a list of that disks' programs, and the screens those programs are on. Unfortunately, one usually has to dig around on the disk (using the INDEX command) to find the directory screen, but that'll take 5 minutes at most.

A SCREEN is a SCREEN

Note that screen #50 is just screen #50 to Forth. A screen number is nothing more than a reference to a spot on a disk. As a result, suppose you put a disk into drive A, then type 50 LIST. Whatever is in the 50th screen on the disk is typed out — whether it is garbage, or a program. If you then insert a different disk into drive A, and type 50 LIST again, you'll see whatever was on THAT disk's 50th screen.

(Well, almost. In truth, after LISTing a screen, it stays around in memory until the memory buffer is needed for a different screen, or until you tell Forth to clear all of its screen buffers by typing the EMPTY-BUFFERS command. So if you type 50 LIST then switch disks, then type 50 LIST, you'll see the same screen, since Forth still has the first screen #50 in memory. But if you type 50 LIST <switch disks> EMPTY-BUFFERS 50 LIST, then you'll see screen #50 for the two disks. Read a book on Forth. It makes sense, and it IS very convenient for some things.)

And if you put one of those disks into drive B:, and want to see what's on screen #50 of drive B:, you have to ask to see screen #210, since screens 0 - 159 are on A:, and screens 160-319 are on B:, so the 50th screen on B: is number 160 + 50 = 210. Or, use the DR0 and DR! commands:

Setting the Current Drive

Actually, there is an easier way using the Forth command DR0 and DR1, which set the 'current drive' to A: or B: respectively. If you type DR1, then drive B: starts with screen #0 — so you could type 50 LIST to see screen #50 on drive B:. But to access drive A:, you need to type DR0 to reset drive A: to the current drive.

I mention that to highlight the fact that Forth will not keep track of where your GET-A-NUMBER program is, and will not know that when you refer to screen #50, you are referring to GET-A-NUMBER. That's your job.

2) THE MVPFORTH DISKS

Very briefly, then, here's how you get started in MVPFORTH:

Boot up your system.

Put F-00 in your A: drive. Type A:<return> to make A: your current disk.

Put your keyboard in CAPS LOCK mode, since all FORTH has to be in UPPER CASE.

Type COMMAND<return>

*** you'll be in MVPFORTH in a moment ***

When you see the sign-on message, put the F-01 disk in your A: drive (you don't need the F-00 anymore)

DON'T REBOOT YOUR SYSTEM WITH THIS DISK IN YOUR A:DRIVE — it doesn't have your system on it. TO REBOOT, type CTRL-BREAK, take the disk out, put a system disk in A:, and CTRL-ALT-DEL or Power down / power up you system like usual.

With the F-01 disk in A:, type
0 20 INDEX
(you'll immediately see line #0 of each screen from #0 to #20 printed out)

Screen #10 should be the DIRECTORY SCREEN. Type 10 LIST to see all of that screen, and see what's on the disk. On mine, I see

15 = assembler load screen
60 = the editor load screen
110 = utility load screen
125 = supplemental load screen

3) You'll need an MVPFORTH manual to go much further. The Mountain View Press (MVP — get it?) noted above has this. Forth is inexpensive and VERY powerful — go for it.

type batty.doc
Going BATty

-by The Magic Rabbit

Most novices, and many more experienced users, don't fully exploit the standard DOS feature of *.BAT files. They're easy to make, easier to use, and can make your life on the PC a whole lot smoother. They're great for lazy typists like myself, or anyone who is tired of typing in strings of repetitive code all the time. So let's look at these mysterious things for a bit.

In general concept, a BAT file is a collection of DOS commands which are invoked with a single command. You can even pass parameters to the BAT files when you invoke them. (For people familiar with EXEC files under WYLBUR or INTERACT, BAT files are extremely similar.) BAT files are simple ASCII files and, when invoked, appear to the PC as though you had typed in their contents on the keyboard, line by line. BAT files can call any DOS command, any .COM command, and any EXE file. Unless it's the last command, though, a BAT file cannot call another BAT file because the second file will over-write and destroy the first one. Confused enough? Okay, let's walk through a simple example.

One very simple BAT file I have is called CLEANPAS.BAT and is designed to clean up the garbage after I've played around with PASCAL a while. It's contents are as follows:

```
ERASE *.MAP
ERASE *.LST
ERASE *.COD
CHKDSK
```

Once the file is created (see below on "how to"), I invoke it just by typing in "CLEANPAS". As you can see, all the commands are straight DOS commands, including CHKDSK.COM. As you can also see, I tend to be lazy.

You can also pass up to 9 parameters to a BAT file. They are labelled %1 to %9 (%0 is special, but don't ask) and are passed in the order you type them in when you invoke the BAT file. For example, another of the BAT files I use with PASCAL, C.BAT (for compiling my program) is this one:

```
COPY %1.PAS B:
A:PAS1 %1,,;
```

If I just wrote a program called TRIALRUN.PAS, to compile it I issue the command "C TRIALRUN" - which translates within the C.BAT file to this:

```
COPY TRAIALRUN.PAS B:
A: PAS1 TRIALRUN,,;
```

(The trailing ",,;" is purely for internal PASCAL use, so don't fret).

If I wanted to pass two parameters, I would list them in the order I use them within the BAT file. If I had a one line BAT file called SEARCH.BAT which contained the line "DIR *%1.%2" and issued the command "SEARCH B BAS" I would get a list of all files that began with a "B" and had the extension "BAS" (a dumb command, I grant you, but okay for an example.) For those of you heavily into typing, you can pass up to 9 parameters with invoking a BAT command just by stringing them out after the name of the BAT file. In practice, though, you'll probably only use

1 or 2 parameters in most of your files.

Two other feature possible within BAT files you might want to use are REMarks and PAUSE. Any string following the command REM writes onto the screen when that line of the BAT file executes. Unfortunately, under DOS 1.10, the word REM also appears. A way around this awkwardness is the unadvertised feature that a period functions just like REM, and looks much prettier. For example, a file called HITHERE.BAT:

```
REM Hi there, micro-freak,
. Seen any good code lately?
DIR
```

Typing in HITHERE would cause the first two lines to print on your screen eaxctly as they are in the BAT file, followed by a directory of your disk.

The PAUSE feature is good if you want to remind yourself to swap disks. It suspends all operations until you hit any key on the keyboard, and then continues executing. You'd normally use it after a REM statement. You might have a BAT file like CGO.BAT (for compile, go) which looks like this:

```
COPY %1.PAS B:
A:PAS1 %1,,;
. Psssst! Put the PAS2 disk into drive A
Pause
A:PAS2
. Psssst! Put the disk with PASCAL.LIB and LINK.EXE in Drive A
Pause
A:LINK %1,,;
%1
```

When I then typed in "CGO TRIALRUN" the first two lines would execute, I'd get some garbage from the Pascal compiler (hopefully saying there were no errors) and then the following would appear on my screen:

```
pause
. Pssst! Put the PAS2 disk into drive A
Strike any key when ready...
```

I'd then swap disks, hit my favorite key, PAS2.EXE would execute (I hope) and the following would appear:

```
pause
. Psssst! Put the disk with PASCAL.LIB and LINK.EXE in Drive A
Strike any key when ready...
```

I again hit some key (after swapping disks, of course!)
and then, with luck, TRIALRUN would go on to link successfully, creating a file called TRIALRUN.EXE. The final command, a simple "%1" just invokes the TRIALRUN.EXE file I just created.

A final special case.... your AUTOEXEC.BAT file. Tired of having to type in the DATE and TIME each time you boot? As easy solution is to create and AUTOEXEC.BAT file with any command in it - with one option being a single blank line. Magically, you're no longer prompted for DATE and TIME. (Of course, if you really LIKE to type in DATE and TIME, just put those commands [i.e. DATE and TIME] into your AUTOEXEC.BAT file, too!)
The AUTOEXEC file automatically executes each time you boot the system

(including Ctrl-Alt-Del boots) so be careful what you put there!

Creating BAT files is easy since they are simple ASCII files. One way is to do it through EDLIN, which you have with DOS. Another way is with any text editor which has the option of storing things in straight ASCII format. BASIC, unfortunately, doesn't work, even with the ",A" option in your SAVE command, since the BASIC line numbers get in the way. The easiest way to create BAT files I've found, though, is through the DOS COPY command. Issue the command "COPY CON:XXX.BAT" and hit RETURN. The screen looks utterly empty, with nary a prompt, but don't sweat it. Just type in whatever commands you like, one per line, hitting RETURN after each line. When you've done your last command and RETURN, type in CONTROL-Z (that is, hold down the "Ctrl" key while hitting the "z" key - or easier yet, just hit the F6 key.) This terminates the collection process and writes the file XXX.BAT which you can then invoke just by typing in XXX. Since BAT files are typically short, only 4 or 5 lines, if you make a typo and need to edit, it's generally easier to just start all over from the "COPY CON:XXX.BAT" step than trying to edit the original though EDLIN or whatever.

Finally - DOS 2.0:
IBM (or rather MicroSoft) has expanded the possibilities of BAT files in the new version of DOS, in particular by offering such features as IF/THEN logic and GOTO instructions. I°ve read about these but don't have DOS 2.0 to try them out, and I'm skeptical of such claims until I actually test them myself. It looks as if IBM is becoming aware of the usefulness of BAT files and is increasing their power and flexibility. This is one of those things IBM may be doing right. BAT files can be a great programming aid. Try them - you'll like 'em!

Here's a quick little one line Basic program that you may find useful. It turns your computer/printer setup into a line by line typewriter. Run "Basica," load the program name (I called it Typeline.bas), then enter a line of text. Make corrections as needed. When you hit the return (enter) key, your printer will print out that line. This will allow you to address envelopes, etc. without all the hassle of setting up your word processing program.

To enter the program, run "Basica," then type the following:

```
10 INPUT X$: LPRINT X$: GOTO 10
```

Then hit the "F4 (save") key, and type "Typeline". This will save the program under the filename "Typeline."

Thanks to Alan Ballweg for this great one-liner.

Questions or Comments?

Linda deSosa 856-6281
Mike Van Waas, Librarian 325-2507
Kevin Ohlson, Editor 494-2574
Wes Danskin, Treasurer 851-0277

IBM Nat'l Service Ctr. 800-428-2569