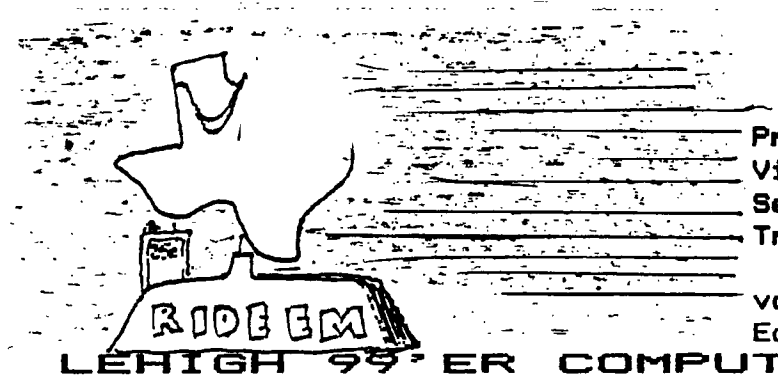

President Mark DeNardo 791-1015
Vice-pres Mike Mattes 252-3468
Secretary Ann Halko 262-8206
Treasurer Mark Appleby 965-3549

hi yo sport!

vol. III, no. 4 April 1985
Editor Frederick Hawkins 432-5913

LEHIGH 99'ER COMPUTER GROUP

Next meeting: Monday, May 20
7:30 PM as usual, we hope.

Community Room, First Nat'l Bank
7th and Hamilton, Allentown

XBASIC:

DATA INPUT

sorting the nefarious widget

In my last article, (December) we explored several ways to sort numbers or strings. As I promised, here is a program to input some data. This routine is for addresses ,but you could use it for other data by changing line 1020 thru 1090 to suit your needs. It is written in Extended Basic which eliminates the continual scrolling that can be disturbing while inputing a lot of data.

address file input routine

pgm by Mark DeNardo

```
10 DIM LN$(100),FN$(100),ST
$(100),CT$(100),ZI$(100),PH$
(100)


1005 ! INPUT ROUTINE
1010 ! ADDRESS FILE
1015 I=1
1020 DISPLAY AT(2,3)ERASE AL
L:"LAST NAME:": : :" FIRST
NAME:": : :" STREET ADDRESS"
: : :" CITY,STATE": : :" ZIP
 CODE": : :" PHONE #": :
1030 ACCEPT AT(4,3)SIZE(-24)
:LN$(I)
1040 ACCEPT AT(7,3)SIZE(-24)
:FN$(I)
1050 ACCEPT AT(10,3)SIZE(-24
):ST$(I)
1060 ACCEPT AT(13,3)SIZE(-24
):CT$(I)
1080 ACCEPT AT(16,3)SIZE(-9)
:ZI$(I)
1090 ACCEPT AT(19,3)SIZE(-24
):PH$(I)
1100 DISPLAY AT(21,1):"PRESS
1-FOR MORE 2-FOR REV. 3 FOR
MENU"
```

(continued on page two-->)

AT THE 10 PORT

The elusive M. DeNardo, long becalmed near the Sargasso Sea, has returned! Bob Wenger, a meteor in our cosmos, goes out in a blaze. His talent shall grace our shores but once -- he's bought a Panasonic. Or was that a Sanyo? Dunno they all PC alike. Ronald Hartranft, saves this issue from p-shaw and me from writer's cramp, delivering the second episode of a survey of Pascal and UCSD p-System. The Sieve didn't this April. Thus, your editor having editted an all-everybody-else issue mounts the aft wheelhouse: Aha! A farce, a recommendation, and assorted denizens of the deep.

April fools and dunderheads

Last issue, page 9: The text on this page was proofread, reprinted and lost. What got printed was the mistakes. The low SYSTEM STATUS byte is at >83FD. This byte when set signifies the soundlist is in the VDP. When >83FD (NOT >83CE) is reset, the soundlist is in GROM. A consistant error in the 'flow chart': the high byte of the SYSTEM STATUS word is the one that adjusts the sound and cursor flash rate. The address is >83FC, not >83FE. The decimal value is -31748, not -31746.

February's assembly language doesn't work atoll. Mea culpa. I simply translated the EQUs from the MINIMEM routines I was using. For some reason, the XBASIC AL requires that you change workspaces. So, an expedient fix for most of the routines would be:

```
MYWS     BSS 32
SAVRTN   DATA 0

NUSTART  MOV R11,@SAVRTN
         LWPI @MYWS
* the routines, adjusted
NUSTOP   LWPI >83E0
         MOV @SAVRTN,R11
         RT
```

Unfortunately, by having to dedicate a workspace, much of the interest in the routines leaks out. A hunch, maybe to be tracked down, is that an effective method would hardcode an out-of-the-way location for the workspace and DATA. Like >A010, say, where XBASIC doesn't use often.

XBASIC: sorted inputs, continued

```
1110 CALL KEY(0,K,S)::IF S=0
THEN 1110 :: IF (K>51)+(K<49
)THEN 1110 :: ON K-48 GOTO 1
120,1030,2000
1120 I=I+1 :: GOTO 1020

2000 !MAIN MENU
2010 DISPLAY AT(1,5)ERASE AL
L:"---- MAIN MENU ----":" --
------------------": " 1 - IN
PUT DATA":" 2 - SORT"
3000 GOTO 3000
```

In this routine we are loading the input data into six arrays. This allows us to compare the array we wish to sort by, then reorder that array. To keep the person's other data with his/hers say last name, we also reorder the other arrays. Suppose we sort by last name LN$, you would insert this routine after the sort routine. Then if the sort routine needs to reorder LN$, we got to this routine. since the "D" sort is the fastest sort routine I'll rewrite it for this address file:

"de_fastest"

```
100 ! "D" SORT
110 ! SORTS 6 ARRAYS BY LN$
WITH N ELEMENTS IN THE ARRAY
120 S=1
130 MN$=LN$(S) :: IMIN=S ::
MX$=MN$ :: IMAX=S
130 FOR I=S TO N
150 IF LN$(I)>MX$ THEN MX$=L
N$(I) :: IMAX=I
160 IF LN$(I)<MN$ THEN MN$=L
N$(I) :: IMIN=I
170 NEXT I
180 IF IMIN=N THEN IMIN=IMAX
190 TEMP$=LN$(N) :: LN$(N)=L
N$(IMAX) ::LN$(IMAX)=TEMP$
191 TEMP$=FN$(N) :: FN$(N)=F
N$(IMAX) ::FN$(IMAX)=TEMP$
192 TEMP$=ST$(N) :: ST$(N)=S
T$(IMAX) ::ST$(IMAX)=TEMP$
193 TEMP$=CT$(N) :: CT$(N)=C
T$(IMAX) ::CT$(IMAX)=TEMP$
194 TEMP$=ZI$(N) :: ZI$(N)=Z
I$(IMAX) ::ZI$(IMAX)=TEMP$
195 TEMP$=PH$(N) :: PH$(N)=P
H$(IMAX) ::PH$(IMAX)=TEMP$
200 TEMP$=LN$(S) :: LN$(S)=L
N$(IMAX) ::LN$(IMAX)=TEMP$
201 TEMP$=FN$(S) :: FN$(S)=F
```

(keep going, top right ↗)

```
N$(IMAX) ::FN$(IMAX)=TEMP$
202 TEMP$=ST$(S) :: ST$(S)=S
T$(IMAX) ::ST$(IMAX)=TEMP$
203 TEMP$=CT$(S) :: CT$(S)=C
T$(IMAX) ::CT$(IMAX)=TEMP$
204 TEMP$=ZI$(S) :: ZI$(S)=Z
I$(IMAX) ::ZI$(IMAX)=TEMP$
205 TEMP$=PH$(S) :: PH$(S)=P
H$(IMAX) ::PH$(IMAX)=TEMP$
210 IF N>S THEN 130
220 ! PUT YOUR RETURN OR GOT
O YOUR MENU/SELECTION ROUTIN
E LINE #
```

Another useful thing to notice in the input routine is the DISPLAY AT command in line 1020. Here I use only one command to fill the screen with my input prompts. DISPLAY AT uses the PRINT separators ', ; :' and TAB(12) just like PRINT and DISPLAY do. One note is to use two :'s in Extended Basic you must type ': :' (a space between them) so ExB doesn't interpret them as command separators. (e.g. 234 A=1 :: B=3)

This covers sorting and data input. In my next article, I'll cover a routine/module to save your data. Then we'll do all the other parts we need to make our program complete.

>M. DeNardo

patching the p-system

The serious errors in the Introduction to PASCAL in the March newsletter (Vol. III, No. 3) occurred in the discussion of wild cards. In the last paragraph of page 3, two "?" were omitted. The corrected sentences follow:

"?.TEXT" is all files on the default device which end with ".TEXT". "A?" is all files beginning with "A".

DFORMAT doesn't with DOUBLE SIDED disks either. If you try you won't get any indication of an error until you try to write to the 181st block. So the moral is: Initialize disks with the Disk Manager module or similar software if you have DS, DD, or both.

Write the following on a slip of paper, and keep it handy while working with the UCSD p-System:

<etx> = <CTRL C>
<esc> = <CTRL .>

>Ronald Hartranft

PASCAL for the TI 99/4A, part II

by Ronald J. Hartranft

the workfile

Two files, #SYSTEM.WRK.TEXT and #SYSTEM.WRK.CODE, are treated specially in the UCSD p-System. (Remember that the prefix "#" stands for the root volume, the disk in drive 1 at system initialization.) The former, if present, is automatically loaded by the editor when you press "E" at the system command level. In addition it is automatically compiled when "C" is pressed at system level to start the compiler. When #SYSTEM.WRK.TEXT is compiled, the compiler automatically stores the p-code version as #SYSTEM.WRK.CODE. One command, "R", works only with workfiles. If #SYSTEM.WRK.CODE is present, pressing "R" will execute it. If it is not present, then pressing "R" will compile #SYSTEM.WRK.TEXT (if present), store the p-code version as #SYSTEM.WRK.CODE, and execute it. In addition, if #SYSTEM.WRK.TEXT has been changed since the last compilation (either "C" or "R"), "R" will compile the new text file and replace #SYSTEM.WRK.CODE by the new p-code version. If there is no workfile present when you press "R", you will be prompted for a text file to be compiled and a name for the p-code version to be created. A workfile by the latter name will be created, and compilation and execution will take place automatically.

There are some convenient file handling commands to deal with the workfiles. Without them, it would be necessary to do a great deal more typing to transfer the work to named files for storage. The filer command, "S", (Save) does a transfer of both files after prompting for a new file name. You enter a name of no more than 10 characters and the system will add the ".TEXT" and/or ".CODE" extensions, as appropriate. If you've used the Prefix ("P") command to make the name of the disk in drive 2 (#5) the default prefix, there isn't much typing to do. The reverse of Save is Get ("G") which will designate the files you choose (.TEXT and/or .CODE) to be the workfiles.

These named workfiles are not very convenient to work with unless you have room in your disk system for both system disks, ED-FILR and COMPILR, as well as a disk for storing your named files. You can accomplish this if you have three drives, of course, or if you have transfered the contents of both system disks to one double sided or double density disk which you keep in drive 1 all the time. The command, What ("W"), gives information on the current workfile, and New ("N") deletes #SYSTEM.WRK.TEXT and #SYSTEM.WRK.CODE and/or removes the workfile designation from the named workfiles. When your programs reach any significant length, you'll want to use named workfiles so that they don't have to be stored on the root volume. In addition you'll want to exit from the editor with the write option if the text workfile is too large for the available space on the root volume.

It isn't necessary to use workfiles at all, but they can simplify your work once you become familiar with the process of using them. However, if you wish to avoid using them, use the filer command "N" to clear out any existing workfiles. Then when you start the editor, you can name the file to be edited (or create a new one) and write it to a named file at the end of the editing session. For compiling, you will be asked for the name of the file to be compiled. If you use "R", you'll be asked for the name of a ".TEXT" file to be compiled and a name for the new ".CODE" file (which becomes the new workfile. You can use "R" to execute this new workfile without recompilation, or you can use "X" to execute any ".CODE" file you name. Note that you needn't (and shouldn't) type the ".TEXT" and ".CODE" extensions.

other UCSD p-commands

In the filer system, we've discussed the workfile related commands, Save, Get, What, and New. Other commonly used commands covered were Zero, Prefix, Remove, Change, Transfer, and List (or Extended list). The remaining Filer commands are Volumes, Date, Make, Krunch, Bad blocks, Xamine blocks, and Quit. VOLUMES lists currently attached devices. In particular it lists disk drives which contain disks and the names of the disks. DATE updates the date stored on the root volume so that your files are stored with the correct date of creation. MAKE sets aside a specified (or default) number of blocks on a disk for a future file. BAD BLOCKS tests for damaged regions of the disk. XAMINE reads a block suspected of being bad until it has consistent data. You can have this data written back to the block to compensate for a possible error in the previous write. QUIT exits from the Filer system to the Command level.

The command I've saved until last, KRUNCH, won't be clear and could be dangerous if you aren't aware of the structure of data on the disk. In the UCSD p-System, a file always occupies contiguous blocks. If a four block file is followed by other files, and is then altered (by editing, e.g.) so that it occupies eight blocks, it will be written in the next unoccupied space available which is at least eight blocks long. The four blocks in the original location will now be unused and available for storing a short file. The original file can be recreated in the original location by using the Make command in the Filer. See the section of the Filer manual on "Recovering Lost Data" (pages 54-57) for some of the methods available. After some time, you will find that your disk has several unused areas interspersed among the files you have stored. The Extended list command shows where they are. The Krunch ("K") command is designed to consolidate all of these unused areas into on big one so that the space can be used for storing larger files than could be put into any of the little pieces. After Krunching, the

Cap'n Krunch, the p-serial continues

previous contents of those scattered areas will be destroyed. Krunching is also somewhat risky if there may be defective blocks on the disk. Files are moved forward on the disk one at a time from the beginning to completely fill all the unused areas, and if a file gets written to a defective area, the result could be merely annoying, or it could be disastrous. Some errors can be fixed by editing -- ".TEXT" files only. A ".CODE" file can be recompiled if the original source code is available.

Backup copies can relieve the anxiety caused by living in a state of tension caused by all the things that can go wrong. You can test disks for bad blocks by using the "B" command. Any blocks suspected of being bad can be extensively tested with the Xamine ("X") command. This will list the files which have data on the blocks you wish to examine and ask for your OK to "fix them?". If you say yes, it will make several attempts to read the data on the block, and, if it manages to read the same thing twice, it will write this data again, presumably fixing it (but perhaps writing erroneous information). If it can't read the data after several attempts, it asks for your OK to "mark bad blocks?". This time, if you say yes, it creates a dummy file on the bad blocks. These dummy files are distinguished by the extension, ".BAD", and are not moved by the Krunch command. However, putting a ".BAD" file in the middle of an existing file destroys the original file.

Manuals which give warnings like the above can cause too much anxiety. In practice things are not as .BAD as they sound. Of my hundred or so disks, only one has ever been defective -- the p-System detected a disk that DFORMAT hadn't done right for a double sided disk. The computer thought that blocks 181-360 (the second side) were defective. Nevertheless, I try to maintain backup copies of everything so that when something does go wrong, I can recover easily. A second disk drive makes maintenance of a backup copy easy for you to do.

I hope you can now find your way around the Filer. Get familiar with the manual so that you know where to look for that detail that you had overlooked before. The details I haven't covered have to do mostly with prompts for file names or warnings that you may be about to cause your own catastrophe. The prompts are self explanatory, but the warnings tend to be rather cryptic. The REMOVE command has the safest logic. TRANSFER is dangerous: if it responds to your input by saying, "Destroy_____?", you've just listed a disk name (no file names) as the destination. Press "N" unless you want to replace the directory of the destination disk with the directory of the source disk. Other warnings ask if it is OK to "Throw away" a file, or to "Remove old_____?".

Other system commands you should refer to in the manual are "C" for the compiler, "I" to reinitialize the system (new root disk or new peripheral connected since booting), "X" to execute a program, and "U" to do it again. For most programming, these commands are simple to use. For advanced techniques you'll need to study all the compiler options as well as the linker ("L") and the assembler ("A").

the UCSD p-System Editor

The editor is invoked by pressing "E" at the system command level. The workfile, if present, will be loaded and displayed for editing automatically. If there is no workfile, you choose between loading some file by name or beginning a new file. To insert new material (into an existing or an empty file), press "I". When you are done inserting, press "<etx>", which means "end of text". You won't find this on the keyboard. Press "<CTRL>", hold it down, and press "C". Most commands in the editor can be aborted by pressing "<esc>", which means "escape". You press "<CTRL>" and "," together to "<esc>". Once you have some text, you should save it periodically by pressing "Q" (Quit). You will have the option of "U", writing (Updating) the text in the workfile, #SYSTEM.WRK.TEXT (# = root volume) and exiting; "E", leaving (Exiting) the editor without saving the changed file (sometimes you don't make any changes--use "E"); "W", putting (Writing) the changed file on a file other than the workfile (and then you have the option to edit some more or exit); or "R" if you didn't mean to press "Q". All editor commands can be displayed on the promptline by pressing "?" except Margin, Set, and Page.

Some of the editing features are like those in the TI BASIC system. The arrow keys are functional for moving the cursor around, and, while in Xchange ("X") mode, the usual function keys, <FCTN 1> and <FCTN 2>, can be used to delete or insert material. For other editing, you will find this editor's "I" (Insert) and "D" (Delete) commands to be very convenient. One of the convenient features available with most commands is the repetition factor. Pressing a number or "/" before a command causes it to repeat that many times or to the end of the file. For example, pressing "1" and "2" and then the down arrow moves the cursor down 12 lines. The spacebar is the same as the right arrow initially. Various other commands, such as Find, depend on what is called the global direction. The global direction is indicated by ">" (forward) or "<" (backward) in the promptline. To change it to backward press "<"; to change to forward, press ">". The spacebar will be equivalent to the left arrow if the global direction is backward. A search for a string can take place in either direction that you choose. "<enter>" moves the cursor to the beginning of the next or the previous line, depending on the global direction which you set.

Cursor Movement

I've already mentioned the arrow keys, <space>, and enter <return> as controlling cursor movement. The Tab key, "<CTRL I>", moves the cursor eight spaces in the global direction, "P" (Page) moves it 24 lines and scrolls the text. The "="

how to get there: blaised trails

also moves the cursor -- to the beginning of the last text inserted with "I" (or found with "F" or replaced with "R", whichever was most recent). To jump, press "J" and "B" or "E" to jump to the beginning or end of the file. You can jump to preset markers put in the text with the Set command, "S".

The best way to move the cursor to a particular point of the text is with the Find ("F") command. Press "F", and you'll see the promptline,

>Find[n]: L)it <target>
or <Find[n]: L)it <target>

where [n] is the number (or "/") you pressed before pressing "F". The "<" or ">" indicates the current global direction--the direction in which the file will be searched from the current location of the cursor. The presence of "L)it" means you are in Token mode and should press "L" to change to Literal mode. In the Token mode, target strings entered won't be recognized unless they appear in the text as complete words bounded by spaces or punctuation marks. In Literal mode, the target string may be a substring of a longer string. I prefer the Literal mode. You should enter the target string you wish to find should be enclosed in delimiters. Most people choose "/", but whatever you choose, as soon as you press the delimiter key a second time, the search will begin. Note that if the delimiter is part of the target string the search will begin before you've entered the complete string. For example, if you respond to the prompt by just typing "/UNTIL/", the search will be in the global direction. The cursor will be left at the character following the "L". A subsequent "=" moves the cursor to the first character of "UNTIL". To find the next occurrence of "UNTIL", press "F" and "S" (for Same).

Text Alteration

The Insert ("I") command allows you to add material just ahead of the currently displayed location of the cursor. "<etx>" (<CTRL C> remember) actually puts this new material in the internal representation of the file being edited. The disk contents are not changed until a "Q" followed by a "U" or "W" is entered. You can also delete ("D") material. Move the cursor to the beginning of the text to be deleted and press "D". Then move the cursor to the end of the material with the arrow keys, "J", or "P". Then if you're sure, press "<etx>". If you're not sure, press "<esc>" (<CTRL .> remember). The Delete command normally stores the deleted material in a buffer so that it can be reinserted at the same or several other locations in the file. More on the buffer later.

The Replace ("R") command looks a lot like the Find command. The prompt after pressing "R" is

>Replace[n]: L)it V)fy <targ> <sub>
or <Replace[n]: L)it V)fy <targ> <sub>

to indicate the direction of search, the repeat factor, [n], and the fact that you are in Token mode and must press "L" to get to Literal mode. The target string will be replaced by the substitution string; both must be enclosed by delimiters as discussed for Find. The Verify ("V") option allows you to veto any substitution by giving you the prompt,

>Replace[n]: R)plce S)ame <esc> aborts

when it finds the target string. Pressing "R" will permit the substitution to take place and go to the next occurrence of the target string (unless it's already performed the command "n" times). Pressing "S" will not make the change (things stay the same), and the cursor will move to the next target. Pressing "<esc>" terminates and skips the remaining substitutions.

The Xchange ("X") command presents you with editing features like those of TI BASIC. You can move the cursor with the arrow keys, replace characters by typing new ones, and insert and delete characters with <FCTN 2> and <FCTN 1>.

There is a "copy buffer" in which text is stored by Delete, Insert, and Zap. The size of the buffer is limited--the limit seems to be related to how much of the total file size of 12800 bytes is used. If you want to put text in the copy buffer for use elsewhere, the safest way is to use the Delete command. If the text to be deleted is too large for the copy buffer, you will get a warning and will be able to press "<esc>". With Zap there is no warning. Zap deletes from the beginning of the last string Found, Replaced, or Inserted to the position of the cursor when "Z" is pressed. This deleted material is put into the copy buffer to the limit of its capacity. Text Inserted ("I") is also put into the copy buffer. The current contents of the copy buffer can be copied to any position in the text by using the Copy ("C") command's Buffer ("B") option. The other option is File ("F") which copies data from whatever file you name to just in front of the current cursor location. Markers placed in the text by the Set command can be used to Copy text if you plan ahead.

Kolumn ("K") and Adjust ("A") are both used to move lines or groups of lines to the left or to the right. You can use arrows or the repeat factor. Refer to the manual for details on these and for the Margin ("M") and Set ("S") commands. These are used primarily for word procesing types of applications.

references

In the next article of this series, I'll present some elementary PASCAL syntax. And we'll start looking at more examples of PASCAL programs. You should buy the reference manual for PASCAL. The compiler manual which TI supplies with the compiler disk describes only the differences between UCSD PASCAL and standard PASCAL. The standard reference to standard PASCAL is the "PASCAL User Manual and Report", 2nd edition, by Kathleen Jensen and Niklaus Wirth, Springer,

practical Pascal books and practice procedures

1974. This will be difficult reading until you've gained some familiarity with PASCAL through this series. There are some textbooks intended for learning PASCAL which you may want to purchase. They are no substitute for the reference manuals, but they can be useful for beginners. A good text is "Programming Microcomputers with PASCAL", by M. D. Beer, van Nostrand, 1982. An alternative by one of the originators of the UCSD System uses turtle graphics routines as the primary teaching tool. However, turtle graphics has not been included by TI in our version of PASCAL. There are alternative text processing types of programs in the book to learn from. The book is "Microcomputer Problem Solving using PASCAL", by Kenneth L. Bowles, Springer, 1977. I recommend this book highly, and if there is sufficient demand I'll supply turtle graphics procedures to be used with the TI 99/4A at a moderate cost.

If you would like another elementary introduction to the UCSD System Editor and Filer, as well as some elementary PASCAL programming, try "Beginners Guide for the UCSD PASCAL System", by Kenneth L. Bowles, Byte Books, 1980.

>Ronald Hartranft

other books

Two references that are found in local mainstream bookstores are published by SYBEX. One, "Introduction to the UCSD p-SYSTEM" by Charles W Grant and Jon Butah has the best map available of the p-System's layout. Although a picture may be worth a thousand, the book retails for $14.95. Its table of contents: Basic Concepts, An Excursion into the System, the Filer, the Editor, Creating Short Pascal Programs, Preparing Large Pascal programs and Appendices. The book additionally contains many sample screens that help you follow its text. This book will be very useful early on and become less needed as you get more practiced.

The second isn't so immediately handy. "The PASCAL Handbook", compiled by Jaques Tiberghien, shows syntax diagrams* for 'Standard' Pascal and five other popular implementations, including UCSD Pascal. Following the brief preface and the remarks, "How To Read This Book" and "How To Read a Syntax Diagram", page one begins an alphabetical list of symbols, predifined indentifiers, and concepts. This is a book to use -- not to read. Thus, it's barely more interesting than a telephone book. And equally indispensible, as it will alert the more-than-casual Pascal user which procedures etc are not found in other dialects, as well as UCSD's peculiarities. The Handbook retailed for $19.95, when I purchased mine; it gets easier to use with practice.

>Frederick Hawkins

*Syntax diagrams are an important element of PASCAL, much in the same way stack diagrams are to FORTH. One suspects that Pascal and its diagrams fall into a subset of chicken/egg situations. For the most part, however, one may use them exactly as one uses TI BASIC's Quick Reference Card -- as a recipe for use.

EXERCISE 2: An Interactive Program.

[The instructions are for a two drive system. If you have three drives, put the compiler disk, COMPILR, in drive 3 (#9). If you have only one drive, refer to the compiler manual, pages 12-14. Use "a" parts the first time with the exercise. Use "b" the second time. If your program was perfect the first time, make a deliberate error (like omitting a ")") and see what happens.]

1. Turn on drives, put ED-FILR disk in drive 1 (#4), put your formatted disk in drive 2 (#5), and turn on computer. If already on, press "I".

2. Press "F", "P", "#5<enter>", "D", enter the date, "Y".

   a. For your first session with this exercise, press "N" (and perhaps "Y") and "Q".

   b. For a later session, press "G", "EX2<enter>", and "Q".

3. Press "E". Editor should respond,

   a. "No workfile". Then press "<enter>".

   b. By displaying the text of EX2.TEXT.

4. a. Press "I" and insert (type) the program, "EXERCISE2", found below. Spacing and indentation are not critical, but be sure to use the same punctuation. Press "<CTRL C>" to accept the inserted text.

   b. Use editor commands to modify the program so that it is like the one found below or to accomplish some other purpose. Feel free to modify the program to suit yourself (and the compiler, of course).

   Note that "<CTRL .>" represents the key for escape, "<esc>".

5. Press "Q","U".

6. Remove your formatted disk from drive 2 (#5) and insert the COMPILR disk in its place.

   a. Press "R", (and wait). If the program works, respond to its prompts. Press "N" if you don't want to repeat the program. If the compiler finds a syntax error, you have the option, if you press "E", to go to the location of the error in the text file and begin editing immediately. You will then be at step 4.

7. Remove COMPILR from drive 1 (#5) and insert your formatted disk in its place.

Continued →

8. Press "F", "S",

   a. "EX2<enter>", and any "Y"s that may be necessary.

   b. If you are offered "EX2" as the workfile name, just press "<enter>", and as many "Y"s as may be necessary.

9. If your program works, press "G", "EX2<enter>", and "Q". If not, just press "Q".

10 You can also run your program now by pressing "R".

11 And as in the last exercise, you can also run it by pressing "X", "EX2<enter>","<space>", "U", "<space>", "U".

12 Congratulations.

Here is the program, 'EXERCISE2".

```
PROGRAM EXERCISE2;
 VAR   S:STRING;
       CH:CHAR;
 BEGIN
  REPEAT
   BEGIN
    WRITELN('   ENTER A STRING OF ',
            'CHARACTERS');
    WRITELN;
    WRITE('   ');
    READLN(S);
    WRITE('   ');
    WRITELN(S);
    WHILE LENGTH(S) > 0 DO
     BEGIN
      DELETE(S,1,1);
```

()editor's note: DELETE is a predefined non-standard procedure found only in USCD Pascal. It will delete specified characters in a string. Syntax is similiar to BASIC SEG$.)

```
      WRITE('   ');
      WRITELN(S);
     END; (* WHILE *)
    WRITELN('   REPEAT? (Y/N)');
    WRITELN;
    WRITE('   ');
    READLN(CH);
   END; (* REPEAT *)
  UNTIL CH = 'N' ;
 END. (* EXERCISE 2 *)
```

that reminds me -

What would you think of a program that you never RUN; a program made up of nothing but REM statements?

Useless, right?

Wrong!! I find it to be one of my most-used programs. I call it "Reminder Calendar".

BASIC programs are ENTERed as a line number and a statement or statements. No matter what order the line are ENTERed, BASIC always sorts them into line number order.

Suppose I wanted to remember the following information:

Wife's birthday is August 15
Anniversary is November 3
Doctor's appointment on February 18 at 11:00 AM
Dentist appointment on April 3 at 2:30 PM
Dinner at Smyths on March 22 at 7:00 PM; bring booze

I would ENTER these items in the following way:

```
 815 REM wife's birthday
1103 REM anniversary
(NOTE! The day is ALWAYS TWO DIGITS!)
 218 REM 11:00 AM doctor appt
 403 REM 2:30 PM dentist appt
 322 REM 7:00 PM dinner @Smitty's; bring Ripple
```

In order to see the entire calendar, type LIST and the calendar will scroll up in date order. If you wish to see reminders for February, type LIST 201-228 (200-300 is OK, too). If you want to see only the next appointment, then type LIST (today's date) and the next appointment will appear. For example, typing LIST 225 might produce:

```
322 REM 7:00pm dinner @Smitty's; bring Ripple
```

Obviously, I've got nearly a month to get the bottle.

In order to ENTER more than one item for a single date, bring the date up with EDIT, then either insert the new information anywhere after the REM or add it to the end of the line. The limit is 255 character per date (line number), so abbreviate whenever possible. Be sure to insert a marker (colon, slash, etc) to separate the items.

You may point to the lines out of range of the days of the month. No month has 40 days, so 340 should be acceptable. LISTing 300-400 for the month of March would then show those items that spilled into the extension.

There you have it... a useful program you'll NEVER RUN!!

>Robert Wenger

programmer's potpourri

XBASIC people rejoice! Courtesy of the LA 99'ers and their Tom Freeman, we now have a public domain version of FORTH that LOADs from XBASIC. Considering that one may SAVE an auto-booting LOAD program, XB_FORTH gets going faster than the more familiar ED/ASM-based version. Otherwise, it runs identically.

Radical Distribution items: The TI network served four new (to us, the rest of the world is 'bout four months ahead) XBASIC programs. Tops on my list is Danny Michael's NEATLISTER which formats a BASIC program to a printer or file. The variable list and line number reference options are its best practical use; they completely replace TI's Programming Aids. The later now look quite flat-footed. (And guess what? They are.)

Back to D. Michael: We also recieved his screendump program which can even be used with Super-Sketch, after you add a LOAD switch to the console. This is also a machine language program that you CALL LOAD/LINK. It will do double-size, etc, and the source code is included so one may recompile for RS232 printers like mine. I haven't yet so my details are pretty sketchy.

Still in the XBASIC radDIST vein, two more disks from John Taylor: A complete SPRITE BUILDER system. His program is mostly XBASIC with some AL routines that speed up the works immensely -- select the Rotate option and your working image rotates, NOW! Not in five minutes, not when you SAVE, not while you wash the car. Ditto for the Inverse video.

SPRITE BUILDER is accompanied by SLIDESHOW, which contains in MERGE format some 100-plus four character (CALL MAGNIFY(4)) sprites. The namesake program will show them all to you. The alphabets, >no kidding<, are especially handsome.

BY THE WAY: PLEASE EXPECT TO MISS ONE OR MORE ISSUES OF THE I/O PORT THIS SUMMER. I finished! a program this month and like to finish the five or six diddley projects begun in previous months. So, taking a page from Craig "now you see him, now you don't" Miller, we are going to a summer schedule.

Speaking of SMART: PROGRAMMER types who rushed out and subscribed after I recommended Miller have finally recieved August '84. Be that as it may, it's time for another BLACK SPOT. The year's magazine is MICROpendium. They're steady, current and publish 30-plus pages for the '4A EVERY MONTH.

Around the beginning of the month, we recieved a news release about the software for the MYARS 99/18. Many have been talking up the hardware details but I'm an advocate for the programmer. So, I figured people might like to hear about the new and enhanced op-codes like WSWS or BLWP2. (WaSh WorkSpace is for sloppy programmers and the other, Bread Loaf and Wurst Platter Too sends out to the corner deli for a sandwich. Structured programmers will like more details on the General Addressing, Relative Branching and Guaranteed Entropy language as will BBS writers the MUltiple Random Protocol Handler with Y-bundling. "Why", I suppose, because we love you. Others may prefer the Assembly Language macros for system operators and developers -- things like RTSP and FFD. The last stands for the built-in protection scheme, Fry Floppy Disk. The powerful ReTurn and Shred Program(mer) is particularly awesome with its optional indirect addressed letter bomb.

Other news to share include the still in-development analog User Input/Output Interface. Unfortunately, until the battery back-up is totally debugged the system is limited to the IRTCL with its useful Chain commands. The companion option to the Interrupt-driven Real Time Cigarette Lighter is still under development -- the potentially useful Auto-servo Coffee IV. (That's IntraVenous, not 4). MYARS's development team is having problems with the feedback loop on the blood sugar/auto sweetner. Until their lead analyst recovers, they're running the Beta (for black) version only.

For the older user, there's the adult 10 Meg Hard--oops!outofspac

>Frederick Hawkins

P.O.Box 4837 * 1501 Lehigh St.
Allentown, Penna. 18103

stamp target
put it
here, pal
Allentown, PA. 18102
PERMIT NO.2018