ESTABLISHED 1979

WEST PENN NEWS

VOLUME 1 NUMBER 7 JUNE 1997

What Ever Happened to... Texas Instruments' Home PC?

by John C. Dvorak

I remember it well. It was June 1979 when Texas Instruments (TI) first rolled out its home computer, the 99/4. It wasn't the machine itself that was interesting, but the fear it generated at the time.

For about a year, there was a buzz about this killer machine. It was the first salvo aimed at the young PC industry. Altair, Apple, IMSAI, Processor Technology, and Northstar were just a few of the companies selling dominant machines in the late 1970s. Once TI made it clear that it was entering the microcomputer scene in a big way, all the small fry began to worry. This would be a test of the industry, a shot across the bow from a company that supposedly knew what it was doing.

The smell of fear permeated the business. Although Commodore and Radio Shack entered the scene with successful machines, the PET and TRS-80 respectively, these companies never generated the anxiety factor that TI did. TI made chips, it knew about computers, and it was huge. Whatever it did would rock the world. And the rumor was that the TI system was going to be a mass-produced 16-bit machine.

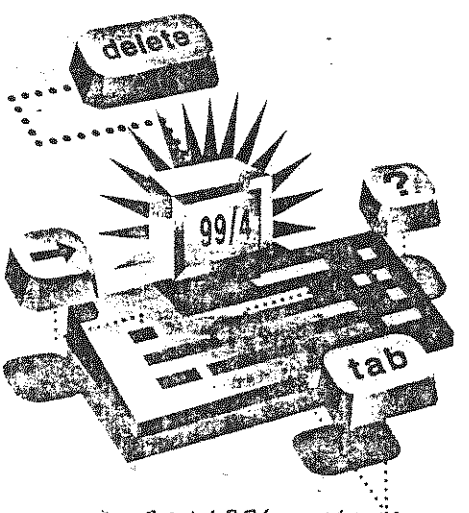
PLAYING GAMES

Everything peaked during the rumor phase, but went downhill after TI released one of the worst machines ever put on the market. By the time it was all over, the company had lost a reported $115 million on trying to make money with that machine. TI would sell an estimated 2.5 million units over the next four years. But most of the sales were lowball $99 sales near the end, and it's hard to prove that anyone really used the thing for anything other than game playing.

Throughout the marketing and sales process for this machine, and even to this day, the real reason the TI 99/4 failed was never mentioned. Now, it will be: The machine had a keyboard with missing keys. There were no question-mark, backspace, arrow, or tab keys. Essentially, the machine was useless for any known computing purpose. I was stunned.

This system was released into a microcomputer market in 1979 that had transformed into a minicomputer model. You'd buy a box with a processor and memory, and to it you'd attach a dumb terminal. These terminals had all the keys. No serious computer user was even going to consider this Texas Instruments fiasco. The home users were the targeted suckers. Even worse, the TI 99/4 was a cartridge-oriented machine initially with no mass-storage capability, not even a cassette port. To write code for it, programmers needed an entirely different computer, so early development efforts were horrible.

The reviews for the machine were interesting since the reviewers were *all* clueless, never once citing the keyboard inadequacies. When the second iteration of the machine arrived, the TI 99/4A, nothing really changed.



WEST PENN 99'ers NEWS

PC WEIRDNESS

The 99/4's chip was designed around a TI/TMS-9900 microprocessor, which was a dud in the marketplace. Some people believe this computer was an effort to get rid of those chips. When the project to design the machine began, it seemed promising. But then CEO J. Fred Bucy decided that R&D should be moved to Lubbock, Texas, where he coincidentally lived. While it made his life easier, far too many engineers were reluctant to relocate to Lubbock, a boring 'burb.

The computer was released at the June 1979 Consumer Electronics Show for a retail price of $1,150, which included a 13-inch color monitor. It took almost a year, though, before it shipped in quantity. This gave people time to notice the weird keyboard and the fact that the machine had no RS-232C interface, no expansion memory, and no cassette I/O. I was told by some old-timers at TI that the machine was designed to be a game machine, hence the cartridge orientation. This made it hard to explain why the joystick was also an option.

Texas Instruments then further muddied the waters in 1979 by announcing the TI 99/7, a $5,000 business computer that never shipped; it was also based on the unpopular TI 9900 microprocessor. Apparently, internal politics at the company killed the machine in favor of the 99/4 home computer. By January 1980, after still more positive reviews by people who should have known better, the machine was selling at the rate of 1,000 units a month—a clear loser. About 30 software packages were written for the system. But it was impossible to develop software on the machine itself, and worse, TI decided to lock out third-party developers by patenting certain aspects of the machine and requiring licenses for anyone selling a commercial product for use on the computer. The company wanted all such profits itself. As a consequence, there were no profits.

IT'S ALL ABOUT PR

Meanwhile, the company went on a PR tear, highlighted by the

continue Page

WEST PENN 99'ERS CLUB INFO

Next Meeting Date: June 17, 1997

Meeting Location: Penns Woods Civic Association

Just off Route 30

N. Huntingdon, Pa

Time of Meeting: 7: P.M.

GENERAL ITINERARY OF OUR CLUB'S MEETING

6:45 P.M.	Doors Open
7:00 P.M.	Genrral Meeting
7:45 P.M.	Demos and New Info
8:45 P.M.	Questions and Answers
9:30 P.M.	One on One Help
10:00 P.M.	Socializing
10:00 P.M.	Doors Close

MEETING HIGHLIGHTS FOR THIS MONTH

Mermaid.....Demo by Paul Brock

PARSEC.....Demo by Paul Brock

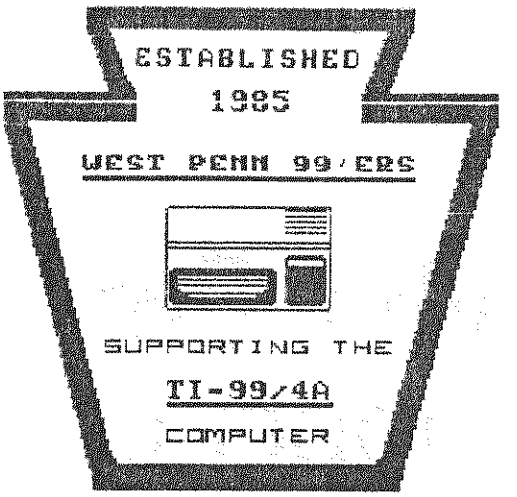
BEYOND PARSEC.....Demo by Paul Brock

Open Interest.....Demo by Anyone

Open Intrest.....Demo by Anyone

LIST OF WEST PENN OFFICERS FOR 1997

President:	Paul Brock	412-478-2754
Vice-President:	Norm Rokke	614-264-6442
Treasurer:	Ed Mandich	412-824-5566
Recording Secretary:	Paul Brock	412-478-2754
Corresponding Secretary:	Paul Brock	412-478-2754
Librarian:	Mickey Cendroski	412-265-5201
Newsletter Editor:	Paul Brock	412-478-2754
Assistant Editor:	Paul Brock	412-478-2754



The West Penn 99'ers Users Group is a Non-Profit organization, dedicated to encouraging the continued use of the TI-99/4A home computer.

Our Membership Fee is:

* $15.00 per year for an INDIVIDUAL / FAMILY membership.
* $10.00 per year for a NEWSLETTER ONLY membership

Those having Full memberships are entitled to the many extra benefits our club has to offer.

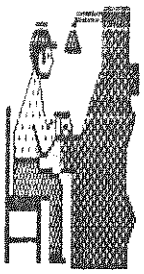
Some of those benefits are:

* Getting to meet some of the nicest people.
* Demos of the latest TI-99/4A software.
* Free copying of our West Penn 99'ers Disk Library.
* Up date of T.I. news,Local,National,International.
* One on one help / Problem solving.
* Participation in our Module Lending Library.
* Participation in our Video Lending Library.
* Ribbon re-inking- for just $1.00 per ribbon.
* Various Computer supplies - at a substantial savings.
* Ability to trade or sell computer equiptment,or electronics.
* Help on getting equiptment fixed.

We meet the third Tuesday of each month at the PENNS WOODS CIVIC ASSOCIATION in North Huntingdon,PA. at 7:00 P.M.

If you can't make it to our meetings...at least become a Newsletter member - and enjoy our NEWSLETTER FORMAT- done entirely on a TI-99/4A computer.

SEE PAGE 10 FOR OUR WEST PENN MEMBURSHIP APPLICATION.



FOR THE RECORD BY PAUL BROCK

MAY MINUTES

It was late when I called the meeting to order. I was wainting to see if everyone got the green light, to come to the meeting. As you all know I had to make a quick cange of dates. Everyone must have got the message. We had a nice turn out.

There was no corrections for the May meeting. The May minutes will stand as read.

The Vice President and Jim Wiegand couldn't make this meeting. Jim's son was in the hospital for heart surgery. I called Jim and his son is doing fine. I also read a post card from W.C. White from Texas. It seems that he is interested in joining the WP.99'ers. I also got a note that Gary Kuehn had some free TI items. I had mentioned this in the April issue, I don't know if he still has them or not. Norm Rokke is looking for a few Micropendiums. June 84- Dec. 84- July 85- Aug 85- Sept. 85- Oct. 85- Nov. 85. Give him a call.

We had a good Treasure's rport, we also got some more 5 1/4 disks DSDD. We are well stocked. Mickey spoke of the library. We will have more information at the next meeting.

Art reported that he had no ribbins to ink, it was due to the date change. We had coffee and both TI systems up and running. We have a monitor If we could get some one to go to Industry Pa. and visit with Lew king and pick it up for us.

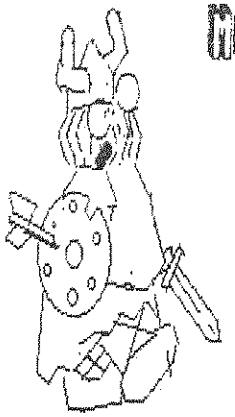
For the raffle prize, a set of coke cola pencles and two 5 1/4 storage cases.

The demo of Munchman and TI Invaders sparked some interst because I didn,t use joy sticks. Mickey is going to try to beat my scores.

Some time something old is new.

Untill then my QUILL has run out of ink!

See you on the 17th of JUNE



MESSAGE FROM THE PRESIDENT & EDITOR

I am sitting here reading an artical from the Computer Shopper, and was appall to its contents. It was by John C.Dvorak. I belive that Mr. Dvorak had some glair from his monitor when he read the history of the 99/4. I didn't see the web site nor do I get it. I do have some of Bill Gaskill's articals and assume that he uses the TI. I will reprint the artical and you can make your own judgement. I belive that TI was so advanced that if it would have stuck with the TI it may have been ahead of IBM. I don't know much of what went on before I bought my TI in 1984,but it seems like I'm up with the 286's. With people like Bud Mills that are still advancing with many additions,no telling how far the TI will go.

At the last meeting Ed was having trouble with a disk. we were tring to use the Disk Fixer, if any one knows how to work the Disk Fixer maybe you can show a few of us how it is done.

TI GAME CARTRIDGE TRICK

I found a trick for most of the games made by Texas Instruments for the 99/4A. This trick allows one to change the starting level. To do this, simply select the game (after inserting the cartridge) and when the title screen of the game appears, quickly type(holding down the shift key) 838.I know this works on the games Moonmine, Alpiner, Munchman, Hopper, and Munchmobile, but those are only the ones I know of, so try and see what other games this works for. This trick is used to change the level of difficulty and some other conditions of a game at the very beginning. This is ideal for advanced players who wish to skip over the simple lower levels of an arcade game. It is also useful for less skilled whho have never been able to reach the upper levels of these arcade games,and want to satisfy their curiosity.

HAPPY BIRTHDAY! TI'er where ever you are!

Tech Section

machine's appearance on the *Mike Douglas Show* in 1981, along with the pop singers The Captain and Tenille. Sales improved, but not enough to make money. This was exacerbated by production problems with the Extended BASIC software cartridge. Lots of games were released, and the machine was kept on the market. Price decreases continued throughout 1982, and the company finally brought out an expansion box for peripherals. But the price of the box, which included 32K of memory, a drive controller, floppy drive, and serial port, was posted at $1,474.75. Plus, the 99/4s were hard to come by even if you could afford them.

By April, the 99/4A cost only $329.95. In June, actor Bill Cosby became the company pitchman for a cool $1 million a year. Meanwhile, the cheap Commodore VIC-20 was taking over TI shelf space. TI then pushed its price down to $299 and offered a $100 rebate. It began to look like a fire sale. Atari and Commodore lowered their own prices even more.

More cartridges were released for the 99/4, along with spreadsheet and accounting programs. The company continued to lose money on the machine. But sales were up! The rebate offer boosted sales as it was extended, and TI made the claim that the 99/4 was the number-one home computer in the U.S. To fight the company, Commodore lowered its price on the VIC-20 to $125, and TI was forced to go lower. No profit could be made at all. In 1983, shipments were halted as a defect was uncovered. The company lost a reported $50 million fixing the problem—a faulty power supply. Commodore was then selling the VIC-20 for $99, and consumers flocked to it.

THE SLIDE

At the 1983 annual stockholders' meeting, TI reported that it had sold its one-millionth home computer. That was the high point for this product. New offerings then caused retailers to start returning their machines. Also, the IBM PC revolution had begun, and the game business was under pressure. The TI 99/4A was selling for $99 by the end of the year. And the company began threatening unlicensed third-party developers, causing an uproar.

The slide began in earnest as critical reviews finally appeared. In January 1984, the company reported that it had sold 2.5 million units. Only 250,000 had any expansion capability. J.C. Penney's dropped the machine the next month. On March 28, 1984, the last 99/4A was produced, and the computer was discontinued. Overall, it was an exercise in futility.

Special thanks to Bill Gaskill for posting an outstanding time line of the 99/4's history on the WWW. It was used as the official chronology for this article. ▼

WEST PENN NEWS

C for yourself

part 3

Norman Rokke

Things got very hectic at the time that the last two newsletters were published and I didn't get an article done. I can't guarantee that I'll have an article every month, but I'll do my best as long as I have something to write.

Last time I said that we would look at writing a C99 program for providing character definitions for centered text of odd length. Before getting to the code for doing this, let's consider how this can be done.

Let's use "ODD", the same text string that we used last time. We saw that the patterns for these three text characters would look like this.

0	0	0	0	0	0
7	C	7	8	7	8
4	4	2	4	2	4
4	4	2	4	2	4
4	4	2	4	2	4
4	4	2	4	2	4
4	4	2	4	2	4
7	C	7	8	7	8

We also saw that the patterns for the four characters which we wanted to produce would look like this.

0	0	0	0	0	0	0	0
0	7	C	7	8	7	8	0
0	4	4	2	4	2	4	0
0	4	4	2	4	2	4	0
0	4	4	2	4	2	4	0
0	4	4	2	4	2	4	0
0	4	4	2	4	2	4	0
0	7	C	7	8	7	8	0

Let's think of the first collection of patterns as a string with 48 hex characters numbered from 0 to 47. The reason for starting at 0 will be explained a bit later.

0	1	16	17	32	33
2	3	18	19	34	35
4	5	20	21	36	37
6	7	22	23	38	39
8	9	24	25	40	41
10	11	26	27	42	43
12	13	28	29	44	45
14	15	30	31	46	47

Likewise, the collection of patterns for the redefined characters can be thought of as a string of 64 hex characters numbered from 0 to 63.

0	1	16	17	32	33	48	49
2	3	18	19	34	35	50	51
4	5	20	21	36	37	52	53
6	7	22	23	38	39	54	55
8	9	24	25	40	41	56	57
10	11	26	27	42	43	58	59
12	13	28	29	44	45	60	61
14	15	30	31	46	47	62	63

If you look carefully you can see that the hex characters which are in even numbered positions in the first grid are in odd numbered positions in the second grid. More specifically, each hex character in an even position of our original string is moved forward 1 position. The character at position 0 winds up at 1, the character at 2 winds up at 3, the character at 16 winds up at 17 and so on.

What happens to the characters at odd positions? Well, the character at 1 winds up at 16, the character at 3 winds up at 18 and so on. Each character in an odd position in the original string winds up at a position 15 higher in the second string.

One last detail must be noted. The first 8 even positions of the second string must be assigned the hex character 0 as must the last 8 odd positions in this string.

This gives us the method for doing the conversion. Starting with the original text string, get the hex string for each text character in order, and join all of these together into one long hex character string.

Then, create a new hex string by moving the characters at even positions in the original string one position forward in the new string. Move characters in odd positions 15 positions forward in the new string. Fill the first 8 even positions of the new hex string with character 0 and do the same with the last 8 odd positions.

Then break up the new hex string up into 16 character strings to be used for redefining characters.

Now that we have a method for accomplishing our task, we can turn our attention to the code. When we do that we encounter a problem. In C99 there is not an available function which does what CALL CHARPAT does in XB. Well I guess that does it for this column.

Wait a minute. I could run an XB program and print out each of the character definition strings obtained from CALL CHARPAT and type them in this article. Well, I could. But I'd probably make at least one mistake, and you might also make a mistake when typing from this article. That doesn't sound like a very good solution.

Hey! I just had another idea. We can get the character definition strings from XB. The C99 program that needs them is a DV80 file. We can produce a DV80 file from an XB program. Let's just write an XB program to produce a DV80 file with the C99 code needed to bring in the character definitions.

Below is an XB program that will do this. Type this program in and run it. Notice that the C99 program file will be produced on DSK2. If you want produce it on another drive, change line 100. I'll discuss how the C99 code works a little later.

```
100 OPEN #1:"DSK2.CENTODD2;C
",OUTPUT,DISPLAY ,VARIABLE 8
0
110 CH=32
120 CALL CHARPAT(CH,A$)
130 CALL CHARPAT(CH+1,B$)
140 CALL CHARPAT(CH+2,C$)
150 CALL CHARPAT(CH+3,D$)
160 PRINT #1:" strcpy(cd,"&
CHR$(34)&A$&B$&C$&D$&CHR$(34
)&");"
170 FOR CH = 36 TO 124 STEP 4
180 CALL CHARPAT(CH,A$)
190 CALL CHARPAT(CH+1,B$)
200 CALL CHARPAT(CH+2,C$)
210 CALL CHARPAT(CH+3,D$)
220 PRINT #1:" strcat(cd,"&
CHR$(34)&A$&B$&C$&D$&CHR$(34
)&");"
230 NEXT CH
240 CLOSE #1
```

Before we look at the rest of the C99 program, you may want to copy to the program disk the other files that will be needed. These include CSUP, GRF1, and PRINTF from the libraries disk. You will also need STRINGFNS from that disk. From the utilities disk you will need SEGSTR/O.

Below is the C program. Load the file CENTODD2;C into the editor and add the additional lines to what was produced by the XB program. The bold text is the code to be added.

```
#define NULL 0
extern grf1(),printf(),chrdef(),hchar(),segstr();

main()
 {
 int length,i,j,k,offset;
 char cd[1537];
 char text1[32],text2[33],hex1[513];
 char hex2[529],str[17];

 grf1();

 strcpy(cd,"0000000000...
 ...
 ...
 ...
```

```
strcat(cd,"0010101000...

strcpy(text1,"ODD");
length = 3;
i=0;
for(j=0;j<length,j=j+1)
 {
 offset=16*(text1[j]-32);
 for(k=0;k<16;k=k+1)
  {
  hex1[i]=cd[offset+k];
  i=i+1;
  }
 }
hex1[i]=NULL;
hex2[i+16]=NULL;
for(i=0;i<16*length;i=i+1)
 {
 hex2[i+1]=hex1[i];
 i=i+1;
 hex2[i+15]=hex1[i];
 }
for(i=0;i<16;i=i+2)
 hex2[i]='0';

for(i=length*16+1;i<16*(length+1);i=i+2)
 hex2[i]='0';
strcpy(text2,"chrdef(nnn,\"");
for(i=0;i<=length;i=i+1)
 {
 segstr(hex2,str,16*i,16);
 chrdef(128+i,str);
 text2[12]=NULL;
 strcat(text2,str);
 strcat(text2,"\");");
 locate(i+3,1);
 printf(text2);
 }
for(i=0;i<=length;i=i+1)
 hchar(1,15+i,128+i,1);
locate(2,15);
printf("EVEN");
locate(23,1);
 }

#include "DSK2.STRNGFNS"
```

After saving the program you may want to create the file for the C-Loader. This file would contain the following.

```
DSK2.CENTODD2;O
DSK2.CSUP
DSK2.GRF1
DSK2.PRINTF
DSK2.SEGSTR/O
```

When you compile this program you need to choose y for the third option Assume long jump for it to compile and assemble properly.

Now let's look at the code. The first line provides information to the compiler. It instructs the compiler to replace the text string NULL with the value which follows in the #define directive, namely '\0'. This value represents the character whose ASCII code is zero.. In C, individual characters are designated by enclosure between single quotes. The backslash followed by a number is used for those characters which can not be readily typed from the keyboard.

We are going to use the ASCII zero character in our program and NULL is also used in the code in file STRINGFNS which we will use.

In the extern statement, we see some new functions. The segstr() function is from Bruce Harrison's utility programs and provides the same capability as SEG$ in XB. The function hchar() works like CALL HCHAR in XB. Both of these are in GRF1.

The first four lines in main() are variable declarations. In C there is only one rule for naming variables. The name must start with a letter and the remaining characters may be either letters or digits. You can not tell the type of data stored in a variable based on its name as you can in XB. You must associate each of your variables with one of the available data type.

In C99 the available data types are integer, character and pointer. Integer

variables can contain any whole number in the range from -32768 to 32767.A character variable can contain any individual character (letter, digit, punctuation etc.) with ASCII value from 0 to 255. We will not discuss pointer variables at this time.

You may also work with arrays of integers or characters. In C99 arrays are limited to one or two dimensions.

In our program we are declaring 5 integer variables. All of these are listed in the same type declaration. We also need several character array variables. In C character arrays are used to represent string values.

In C strings are terminated by an ASCII zero character. When setting the size of a character array to be used for string data, you must remember to leave room for the ASCII zero. The variable cd is a character array we will use to store the hex string for defining all of the characters from 32 to 127. This requires 16*96 or 1536 characters. We must set the size of the array at 1537 to allow for the ASCII zero. The variable text1 will be used to store the string of text characters to be centered (an odd number of characters with maximum of 31). We will use hex1 to store the hex definition strings of the characters in text1 (maximum 512 characters). In hex2 we will store we will store the hex definitions of the redefined characters. The hex definition of a single character will be temporarily stored in str. Variable text2 will be used to hold and print on screen the C99 character definition statements needed for the redefined characters.

When using arrays in C it is important to remember that the first position in the array always has index 0. This is why we started numbering the cells in the grids at zero. For example str would have valid indexes in the range from 0 to 16. The number inside brackets in the declaration is the number of values that can be stored.

It is also important that you realize that the compiler does not check to see that array indexes are in the proper range. For example a reference to str[37] would not prevent your program from compiling and assembling. However, it probably would cause some problem somewhere and worst of all the problem would show up as something totally unrelated to the source of the problem making it very difficult to debug.

C puts a lot of responsibility on the programmer to handle details that can be ignored in XB. The advantages of the language don't come without cost.

We begin by putting the computer in graphics mode. Next we come to the code produced by the XB program. This code creates the string cd. The function strcpy() performs the same task that CD$="ABC" would do in XB. It lets us assign a value to a string variable. It provides the ASCII 0 for string termination also. The strcat() function does what CD$ = CD$&"DEF" would do in XB. It also takes care of the ASCII 0 terminator. The result is that the string cd contains the hex character definitions of all of the characters from 32 to 127.

Next we assign "ODD" to text1 and set length to 3. Note that assignment of a value to an integer variable is done just as in XB.

Now we need to get the character definitions for each of the characters in text1. This is done by a nested pair of for loops. We use i as an index to the position in the hex string hex1. This is set to zero before the loops.

The for statement is followed by three expressions separated by semicolons. The first expression gives an initial value to the loop control variable. The second expression is used to determine when to stop the loop. As long as this expression is true, the loop continues. When it becomes false, the loop ends. The last expression is

performed at the end of each pass through the loop. In this case the loop starts with j=0 and continues for all values of j up to length-1. These we use as index positions to all of the characters in text1 which actually have data (0 to 2). The last expression takes care of increasing j by one each time through the loop.

For those of you who may know some C, I know that the incrementating of the loop control variable is not typically done like I have done it here. However, I wanted to keep things as simple as possible at the beginning. I will discuss the more typical way of doing this in a latter article.

Variable j is an index to a position in the string text1. The outer loop takes us through each of the positions in the string in text1.

Character data is stored as ASCII code values so text1[j] will be the ASCII code of one of the characters in the string we want to center. If j is 0, text[j] will be 79 (for 'O'). By subtracting 32 (ASCII code for the space character), we determine how many characters come before the one we are dealing with. Multiplying by 16 gives us the position in cd where the definition of the current character begins.

The inner loop runs from k=0 to k=15 to get each of the 16 hex characters that define the current text character. Inside the loop we also increment i.

Also notice the use of curly brackets to create a block of statements , all of which are done each time through the loop. Indentation also is used to make it easier to tell what is done inside the loop.

Next we put the ASCII zero character into hex1 and hex2. Variable hex2 is 16 characters longer because it defines one more text character than hex1. Since we are not creating these strings by using string functions, we are responsible for making sure the ASCII zero is in the proper place.

The next loop moves each character in hex1 to its proper position in hex2 as we discussed earlier. Characters at even indexes are moved ahead 1 position and characters at odd indexes are moved ahead 15 positions in the new hex string (hex2).

The next loop fills the first 8 even index positions in hex2 with '0'. Then the next loop fills the last 8 odd index positions in hex2 with '0'. Note that this is the character '0' (the same as "0" in XB) and not '\0'.

Next we redefine characters and print the appropriate C99 statements to do this on the screen. Outside the loop we define the first part of the line of text to display on the screen. Note the \" in the string that is to be assigned to text2. This is how we can put the " character inside a quoted string. The \" becomes only one character. It's comparable to using double quotes in XB to get one quote character.

Inside the loop we use segstr() to get individual character definition strings from cd. Then we redefine a character. Next we put the ASCII zero at the end of text2. The first time through the loop it will already be there since we used strcpy() outside the loop. Inside the loop we're going to add to the string. After the first time through the loop the ASCII zero will not be in this position so we have to put it there so we can add to the same basic string each time through the loop.

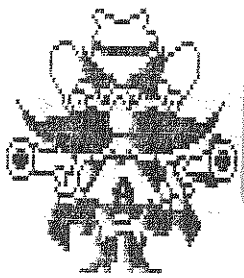
Next we add to text2 the hex string character definition. Then we add "); to the end of the string. Using the locate() function which we saw last time we position the cursor at column 1 of a particular row and then print the chrdef statement to the screen.

In the next loop we display the four redefined characters centered on the first row of the screen. I have not been able to use printf to deal with characters above 127 so I used the hchar() function. This is a function included in GRF1 which works

CONTINUE NEXT MONTH

WEST PENN 99'ERS
C/O Paul A. Brock
General Delivery
North Apollo PA 15673-9999

FIRST CLASS MAIL



MICKEY CENDROWSKI
100 PINE STREET
RUSSELLTON, PA 15076
USA

FIRST CLASS MAIL

This newsletter was composed in it's intrety
using a TEXAS INSTRUMENTS TI-99/4A computer

NEXT MEETING JUNE 17th. 7:00 PM

UP MEMBERSHIP APPLICATION

Name_____
Address_____
City_____ State_____ Zip_____

Please check one

☐ Newsletter membership
No library privilages
$10.00 Per year

☐ Full membership
Library privilages
$15.00 Per year

Complete and return to

Ed Mandich, 304 Chicora St. East McKeesport,PA.15035