TEXAS INSTRUMENTS

The Inner Limits

by Glenn Davis

When TI released its second version of TI-Writer to user groups, it included an additional file named Charal. The purpose of Charal is to give the TI a "real" lower-case character set instead of the squashed capital letters normally used.

Until I discovered what those characters actually looked like, I was anxious to get the update. Some characters, the asterisk and zero among them, looked really bad. The zero looked like "Gumby" with a lump on one side and the asterisk looked like a squashed insect, or maybe "Pokey" on a bad day. On a color monitor, the small "e" and "s" were hardly readable. It was almost as if TI wanted people to say "oh yuck!" and find out how to write their own.

Well, I slept on that idea, wondering how to modify the Charal file TI graciously sent us. Then, one day while I was playing around with a debugger searching GROMs of various cartridges to see what I might find, I discovered that the name Charal was in TI-Writer's GROM, not on the program disk. The implication of that was immediately apparent. Any TI-Writer cartridge will try to load Charal. TI-Writer subsequently loads the editor files, whether Charal is found or not. This is why the character set changes before the editing screen appears. Somebody at TI was really thinking when they designed that feature.

So, the Charal file is just a memory image file that is loaded where the Pattern Descriptor Table is in VDP RAM. Since this knowledge has been out for almost as long as the update of TI-Writer, many other programs also use Charal. Fast-Term by Paul Charlton is one example (you HAVE sent Paul some bucks for this fine "freeware," right?). This allows both the terminal program and the word processor to reside on the same disk and share Charal.

The files at the end of this article will allow you to create and modify your own version of Charal. Names of these files will be a little sticky, so please bear with me. Two different character sets are presented in the article. The first set will be referred to as the "TI" set. This set is very similar to the Charal set that TI sent out. I have made modifications to some characters, such as the numeral zero, which now includes a nice slash to allow easy distinction between it and the letter "O." The asterisk and some other punctuation characters are improved too.

The second character set I will refer to as the "Apple" set, since they are similar to the characters used by the Apple IIe and IIc computers. In this font, all characters are fully formed. No squashed "e" or "s." The capital letters are seven dots high. When a decending lower-case letter is above a full upper-case letter, there will not be any space. They appear this way on the Apples too. Having a decender above a capital letter is rare, though. The prime advantage of this character set is that the letters are large enough to be readable on a color monitor.

Before I tell you how to assemble these files, let's examine what other uses there are for "optional" fonts. Besides modifying characters at will, these fonts, can be used to create international fonts. My FX-80, for example, has nine different countries' character sets that can be selected by software. In each of them, selected punctuation characters are replaced by their foreign counterparts. With the German set, "{" becomes "ä" and "|" becomes "ö." By making appropriate adjustments to the TI of Apple fonts, these characters will actually appear as such on your screen. If you regularly use foreign fonts, placing small stickers on the keys allows easy access to these characters. Some printers allow "custom" characters, so just about any characters can be developed, even the Greek set.

Those of you with daisy wheels can do similar things. Diablo wheels (as one example) come in German, French, Norse, British, and Spanish. The instructions should indicate which ASCII codes print the foreign characters. Additional wheels may be available for other languages such as Russian, so check with your dealer.

Another way to make use of these fonts is to modify the control character definitions. Currently they are the ones used by TI-Writer. The <CR> symbol can easily be changed to a paragraph symbol or a broken arrow or anything else you'd like. The ASCII 08 character (backspace) can be changed to a reverse arrow (this character is accessed by pressing CTRL-U; SHIFT-H; CTRL-U).

Sometimes inverse characters are desirable for screen display, as when punctuation characters must match (braces, brackets, quotes, parentheses). Even making all of the control characters inverse may have benefits in certain situations. If you upload a legal character set (to a capable printer) or use a legal daisy-wheel, adding the copyright, trademark, and other legal symbols to any fonts would be ideal. In text-mode programs such as TI-Writer, inverse characters have their dot patterns reversed. So where an "on" dot normally is, put an "off" dot. And vice-versa. Technically, only the six left-most pixels are displayed, but inverting all of them has no ill effects.

Some people find TI-Writer's "features" are bugs. A frequently cited "feature" that people dislike is that TI-Writer puts two spaces after every period. Using alternate fonts, this feature is easily circumvented. Merely redefine some character (such as "|", the vertical bar, little used in most writing) as a period. An inverse period would be even better, making these periods easily distinguishable from the "ordinary" ones. Then (and this is the "trick") use the Transliterate formatter command .TL 124:46 near the beginning of each file. That changes the bar (character 124) to the period (character 46) when the formatter prints it. See the TI-writer manual if you do not understand how this command operates.

To assemble these files, begin with the source files all on one disk. Then load the Assembler. To assemble the TI font, use Charal/S. To assemble the Apple font, use Chara2/S. I use the convention of the TI-font files using a "1" and the Applefonts using a "2." Mixing them is not a good idea because the Apple characters are taller. Some files, i.e. the control character file, are shared. The source files have just Copy directives in them. Make sure the directives call for the files on the drive with the source disk in it. They are set up for DSK1 now. Use Charal/O or Chara2/O for the object files.

When that finishes, load the SAVE utility from the Editor/Assembler disk B and then the object file you chose to assemble through the Load and Run menu option. Press <Enter> again and the loader will ask for a program name. Enter SAVE. The save utility will give you some warning messages and prompt for another filename.

If you intend on assembling both of these fonts, name the TI set Chara-TI and the Apple set Chara-AP. Otherwise enter Charal (which makes it immediately available for use. You can then ignore the disk manager instructions below). When it finishes in a few seconds, use a disk manager to make a copy of the Chara-TI

Program on Page 260
continued on page 261

The Source Hosts New TI-SIG

Source Telecomputing Corporation (STC; McLean, VA)—long a supporter of TI-99/4 users orphaned when Texas Instruments discontinued the model—now offers them a new, much improved "home." The new TI-SIG (special interest group) replaces TEXNET—the Source's pioneering 99/4 on-line user group, established in 1981—and adds a number of significant enhancements. It offers directories of SIG members; user groups; bulletin board systems; and a robust library of public domain software and shareware available for downloading. Registration on The Source Information Network is available for $49.95 (all prices suggested US resale) at participating resellers or direct from Source Telecomputing Corporation. Its lowest on-line charges (up to 50% lower than a leading competitor) are those it now charges for connection to most SIG features; these start at just 10¢ per minute ($6 per hour; $10 monthly minimum).

Source TI-SIG bulletin boards offer information on assembly, BASIC, Pascal, C and other languages; ads; hardware and software for sale; educational items; and shows or fairs of interest to 99/4 users. The TI-SIG library of public domain software and shareware for the 99/4 can be downloaded using XMODEM, TEII and other advanced file-transfer protocol for error-free file transfers. A popular TEXNET monthly on-line column, *Randy's Rumor Rag*, provides information on products, shows and other items of interest to TI-99/4 computer users.

Veteran SysOp (system operator) Blaine Crandell manages and coordinates the TI-SIG. Crandell has served as SysOp for TEXNET since 1981.

The TI-SIG joins a growing number of SIGs already on The Source for users of IBM PCs, Apple II/III and Macintosh™ computers. SIGs for users of Ashton-Tate and Microsoft products are also on-line.

The Source offers a wealth of easily accessible information and communication services. These include SourceMail (electronic mail for members), computer conferencing, a live Chat facility for real-time conversations among users, and more than 65 bulletin boards. The latest news and sports stories and feature articles are available on-line. The Source also offers a complete investor service, travel services, an encyclopedia, on-line catalog shopping, games and more. A unique on-line tutorial and introduction makes it easy for new subscribers to use the services on The Source, free of on-line charges.

For additional information contact Source Telecomputing Corporation, 1616 Anderson Road, McLean, VA 22102; (800) 336-3366.

Mention that you read about it in *Computer Shopper*. ●

The TI Forum

by Ron Albright
& Jonathan Zittrain

Quite a Year...

While I won't do a complete "year in review" piece now. I reserve the right to do one later. I want to look back at a few people who made major contributions to the TI community in 1986. In my eye, the biggest gift we received was from Canada. Clint Pulley's release of c99 made a version (albeit unlimited) of the computer industry's most popular programming language available to the 99ers. Several folks picked up on the gift and gave in return. Notably, my "guru," Warren Agee. Warren abandoned Forth and ran with c99 with a proliferation of helpfiles and code that has taught the TI world how to get c99 rolling. Hardware product of the year? Has to be Gram Kracker—the little box that cracked the GROM cartridge for all to see and hack away at it for the better. Runner-up? Horizon Ramdisk, which gave new meaning to "disk-access speed." Software product of the year? Since I like art, I give this category to Travis Watford who developed the means for the TI to use the hundreds of RLE graphic art pieces from Compuserve. TI users can now download graphics to the TI, import them into our own graphics software (Graphx or TI Artist) and modify them or whatever. Software producer of the year? Asgard Software which reaffirmed what Borland proved in the MS-DOS world—good products at reasonable costs will sell even in the face of rampant piracy. While others blamed falling sales to everything from piracy, track copiers, and disgruntled columnists, Asgard was a shining light of marketing genius. Best Publications? No way to decide this! If you are looking for up-to-date news and product reviews, it's *Micropendium* by a mile. If it's technical information, schematics and a close contact with European product development, it's the *R/D Newsletter* from Ryte Data. And if you want a novel publication with programs, tutorials, graphics and utilities already on disk in a menu-driven format, the *Traveler Diskazine* is an incredible value. I hope I can eat much crow with this (cough) "award," as I am writing it in September, before the Chicago Fair, but I award to Disappointment of the Year to Myarc—how long do we have to wait for the "new computer?" I have, frankly, found increasingly tedious the "R.S.N." ("real soon now") hype. As I said, I will be pleased to try and dine on a case of crow, my hat, and Myarc's doormat if the machine is available to the consumer before the end of 1986.

So there you have it. One observer's thanks and "awards' to those who have made 1986 the best year yet for the TI users. At the end of 1985, I asked myself, "What more could this hearty band of orphans possibly accomplish next year?" Well, guys and gals, you showed me. And I remain impressed. Can 1987 provide still more encores? Knowing the TI programmers and hackers as I do, the answer is a resounding YES!

Delphi is like a domed stadium in Columbus, Georgia. It's a nice facility, and the grandstands are real comfortable, and there is an excellent supporting team that plays there, but it sure gets lonesome being the only one there cheering. Ticket sales are almost nil. At least where the TI community is concerned. That is a shame too. It is an excellent system, with rates cheaper than Compuserve and the Source ($7.20/hour non-prime time which includes Tymnet charges and no surcharge for 1200 baud), no monthly minimum charges, an on-line encyclopedia (available at no extra cost), a gateway to Dialog (one of the largest databases in the world), and an electronic mail system that allows you to send mail to users on Compuserve and the Source. Add to that a well-staffed and run TI SIG (called the "TI Information Network"), and you wonder why they have not attracted more TI users (or users in general for that matter!). It would appear that the answer is, simply, lack of exposure. General Videotex (who owns and runs Delphi) seems to really believe that you have to merely build a good stadium (or on-line service) and people will beat a path to your door. No need to put up billboards or print maps for directions. People will just show up. Well, it hasn't happened that way. While Compuserve and STC spend big bucks on ads, I have never, ever seen an ad for Delphi. Have you? Its too bad too. I just stumbled on to Delphi through a friend and really enjoy the system. TIIN is supported by Dick Evans and Jeff Guide expertly, but they seem to be casting their technical and informative TI pearls upon barren astroturf. They have bent over backwards with a great sign-up offer ($10 for a password, reference card, and an hour usage credit). Here is a great bargain, if you are a subscriber to *Computer Shopper*, Delphi will waive the sign-up fee. Just give them your subscriber number from your label. I took them up on it and plan to hang around there often. Try it out. You don't have a thing to lose. To sign up (at least as of this writing in mid-September), call your local Tymnet number (no Telenet access to Delphi), at the "Please type your terminal identifier" prompt (the prompt is garbled at 1200 baud), enter "A". At "Please log in:", enter "Delphi". At "Username:", try "JOINTI99", and at "Password:", try "Teledata". You can sign up directly there. If this offer is no longer in effect, call 1-800-544-4005. Some day maybe General Videotex will spring for some ads and the world will find the way to the grandstands.

Triton Still Around

Did you get your "Fall '86" Triton Catalog? They still have loads of TI stuff and, in particular, feature some good buys on Corcomp equipment (they, mysteriously, have dropped the Myarc product line) and TI cartridge software. If you didn't get one, write them at PO Box 8123, San Francisco, CA 94218.

c99 Game Coming

In the January issue, I (RA) will be starting a mini-tutorial of a game I wrote in c99 called "CINVADERS" (get it? "C Invaders"?). Depending on how much other news I have to report in the coming months, it will run anywhere from 2 to 4 months—a little each time. It's not that long, but I will try to teach a little as we go along. Hopefully, I can talk my resident c99 "guru." Warren Agee, to make some comments or suggestions on how the code could be improved. Anyway, we'll start that in January.

Converting BASIC to c99

When I started to learn BASIC (and later, Extended BASIC), I remember how I did it. I first typed in other programs from magazines and books. Then I started to do my own programs. And the first type of commands I used were the graphics commands. I sure didn't jump in with file handling or string manipulation! Anyway, I found myself doing the same thing with c99. I typed in some programs out of a book, then started playing with my own routines with graphics. Then I tackled a game. I have thought all along that if you can learn the logic involved in a game, you have learned a great deal about the programming structure of a particular language.

In this tutorial, we will try to accomplish a couple of things. First, a glimpse at some of the graphics commands available to c99 in the "grf1rf" library (that comes with all version 1.32 or higher), and, secondly, a look at how to convert a short BASIC graphics display to c99. It really isn't that hard.

Listing 1 is a short BASIC program from Ed York that has appeared in several UG newsletters. It is a colorful graphics display. Listing 2 is a conversion of the program to c99, done by me. They both accomplish the same thing graphically. I have commented the c99 source code to try and explain step-by-step what we did. I think as you look at the programs, you will see how similar both the graphics commands and the logic is between c99 and BASIC. It is, to me, much closer to BASIC than Forth was. See if you agree.

Notes

[1] Compile the program with the compiler. Make sure the D/V 80 file "RANDOM:C" and "GRF1" is on disk 1, then assemble the output file. Then, load the assembler output (which should be a D/F 80 file), then from E/A option 3 still load the file "CSUP" (another D/F 80 file) and "GRF1" a third D/F 80 file. Then hit enter and use the program name "START". It should run.

[2] The only complicated move was separating lines 220 through line 300 into the separate function "fun()". This was done because line 300 in the BASIC program is a GOTO 220. Since there is no GOTO function in c99, we separate out those lines and use recursion in "fun()". Recursion simply means a routine calls itself over and over, just like a

continued on page 261

LISTING 1

```
100 REM COLOR BONANZA
110 REM WRITTEN BY:
120 REM ED YORK
130 CALL CLEAR
140 FOR A=40 TO 136 STEP 8
150 CALL CHAR(A,"55AA55AA55AA55AA")
160 NEXT A
170 FOR B=2 TO 14
180 CALL COLOR(B,1,1)
190 CALL VCHAR(1,2*8,24+8*8,22)
200 CALL VCHAR(1,2*8+1,24+8*8,22)
210 NEXT B
220 FOR C=2 TO 14
230 CALL SCREEN(INT(16*RND)+1)
240 FOR D=2 TO 14
250 CALL COLOR(D,8,C)
260 NEXT D
270 CALL KEY(0,E,F)
280 IF F<1 THEN 270
290 NEXT C
300 GOTO 220
```

LISTING 2

```
/* COLOR BONANZA This and the next 3 lines are REM's (line 100) */
   WRITTEN BY:                                          (line 110)
   ED YORK                                              (line 120)
   converted by Ron ALbright*/

#include dsk1.grf1rf /* required to use the graphics commands */
#include dsk1.randomic /* required to use the random number commands */

main()
  {
   int a,b; /* MUST declare ALL variables used in a routine at start */
   grf1(); /* MUST be used as first command for graphics library use */
   clear(); /* Same as CALL CLEAR (130) */
   randomize();/* Same as RANDOMIZE in BASIC */
     for(a=40;a<=136;a=a+8) /* Lines 140 and 160 ALL IN ONE STATEMENT! */
       chrdef(a,"55aa55aa55aa55aa"); /* CALL CHAR in line 150 */
       for(b=2;b<=14;b++) /* Another FOR-NEXT loop -lines 170 and 210 in one */
          { /* Multiple lines in for loops need to be braced */
            color(b,1,1); /* Same as CALL COLOR - line 180 */
            vchar(1,2*b,24+8*b,22); /* Just a plain old CALL VCHAR! line 190 */
            vchar(1,2*b+1,24+8*b,22); /* line 200 */
           } /* Closed braces after FOR LOOP */

   fun(); /* Gets a little tricky here. Since there was a "GOTO" statement in
             line 300, I decided to make a new routine starting at where the
             GOTO directs the BASIC program - line 220. That way, I can call
             the second function from itself, in essence, creating a "GOTO".
             See below. Anyway, that is why I started a new function called
             "FUN()". I call it from the Main() routine here by just calling
             the name of the routine. Its just like I said GOSUB or, in XB, had
             created a user-defined SUB FUN and, here, said CALL SUB FUN. */

 } /* All of the main() */

fun() /* Start of a new function */
{ /* All functions start with an open brace */
     int c,d; /* Declare thoses variables at the start!!! */
     for(c=2;c<=14;c++) /* start of another FOR loop-lines 220,290 in one! */
     { /* multiple lines after a FOR need to be braced! */
          screen(rnd(16)+1); /* CALL SCREEN in line 230 */
          for(d=1;d<=14;d++) /* Start of a nested FOR LOOP - line 240 */
          color(d,d,c); /* CALL COLOR in line 250 */
          getchar(); /* Just waits for a key to be pressed - lines 270,280 */
      } /* Close that brace for the FOR loop */

     fun(); /* See that GOTO 220 in the BASIC program? This is the
               same thing - it just keeps calling "fun()" which is nothing
               more than the program starting at line 220.  so, by seperating
               the lines where the GOTO starts into a separate routine, we can
               now call it over and every time we would be using the GOTO in
               Basic. */

} /* Close braces for fun() routine */
```

"Inner Limits" Modified "Charal" Character Sets

CTRLCHAR and PUNCT-D are shared by both the TI font and the Apple font.

CTRLCHAR

```
        DEF SFIRST,SLAST,SLOAD
*                                       ASCII

SFIRST
SLOAD   DATA >0020,>0000,>1824,>2418    0
        DATA >0020,>0008,>1808,>081C    1
        DATA >0020,>0018,>2408,>103C    2
        DATA >0020,>0018,>2408,>2418    3
        DATA >0020,>0014,>141C,>0404    4
        DATA >0020,>001C,>1018,>0418    5
        DATA >0020,>0008,>1038,>2418    6
        DATA >0020,>001C,>0408,>1010    7
        DATA >0020,>0018,>2418,>2418    8
        DATA >0020,>0018,>241C,>0408    9
        DATA >2020,>3800,>1C10,>1C10    A
        DATA >0040,>0020,>2038,>2438    B
        DATA >0070,>5070,>4854,>1C14    C
        DATA >0070,>4070,>001C,>1010    D
        DATA >0020,>0018,>243C,>2018    E
        DATA >0040,>0814,>101C,>1010    F
        DATA >0040,>4040,>1824,>2418    10
        DATA >0020,>2020,>2808,>0808    11
        DATA >0040,>4058,>2408,>103C    12
        DATA >0040,>4058,>2408,>2418    13
        DATA >0040,>4054,>141C,>0404    14
        DATA >0040,>405C,>1018,>0418    15
        DATA >0040,>4048,>1038,>2418    16
        DATA >0040,>405C,>0408,>1010    17
        DATA >0040,>4058,>2418,>2418    18
        DATA >0040,>4058,>241C,>0408    19
        DATA >0040,>4040,>1824,>3C24    1A
        DATA >0040,>4050,>101C,>141C    1B
        DATA >0040,>4040,>1C10,>101C    1C
        DATA >0040,>4444,>041C,>141C    1D
        DATA >0070,>7070,>7070,>7070    1E
        DATA >0040,>4C50,>101C,>1010    1F


PUNCT-D

        DATA >0018,>2020,>4020,>2018    {
        DATA >0010,>1010,>0010,>1010    |
        DATA >0030,>0808,>0408,>0830    }
        DATA >0000,>2054,>0800,>0000    ~

SLAST   END
```

CHARA1/S (copies for TI font)

```
        COPY "DSK1.CTRLCHAR"
        COPY "DSK1.PUNCT-A1"
        COPY "DSK1.DIGITS-1"
        COPY "DSK1.PUNCT-B1"
        COPY "DSK1.UCASE-1"
        COPY "DSK1.PUNCT-C1"
        COPY "DSK1.LCASE-1"
        COPY "DSK1.PUNCT-D"
```

PUNCT-A1

```
        DATA >0000,>0000,>0000,>0000    (SPACE)
        DATA >0010,>1010,>1000,>1000    !
        DATA >0028,>2828,>0000,>0000    "
        DATA >0028,>7C28,>287C,>2800    #
        DATA >0038,>5430,>1854,>3800    $
        DATA >0044,>4C18,>3064,>4400    %
        DATA >0020,>5020,>5448,>3400    &
        DATA >0008,>1020,>0000,>0000    '
        DATA >0008,>1010,>1010,>0800    (
        DATA >0020,>1010,>1010,>2000    )
        DATA >0028,>107C,>1028,>0000    *
        DATA >0010,>107C,>1010,>0000    +
        DATA >0000,>0000,>0030,>1020    ,
        DATA >0000,>007C,>0000,>0000    -
        DATA >0000,>0000,>0030,>3000    .
        DATA >0004,>0810,>2040,>0000    /
```

DIGITS-1

```
        DATA >0038,>4C54,>5464,>3800    0
        DATA >0010,>3010,>1010,>3800    1
        DATA >0038,>4408,>1020,>7C00    2
        DATA >007C,>0418,>0444,>3800    3
        DATA >0018,>2848,>7C08,>0800    4
        DATA >007C,>4078,>0444,>3800    5
        DATA >0038,>4078,>4444,>3800    6
        DATA >007C,>0408,>1020,>2000    7
        DATA >0038,>4438,>4444,>3800    8
        DATA >0038,>4444,>3C04,>3800    9
```

PUNCT-B1

```
        DATA >0000,>3030,>0030,>3000    :
        DATA >0000,>3030,>0030,>1020    ;
        DATA >0000,>1020,>4020,>1000    <
        DATA >0000,>007C,>007C,>0000    =
        DATA >0000,>1008,>0408,>1000    >
        DATA >0038,>4408,>1000,>1000    ?
        DATA >0038,>4454,>5840,>3C00    @
```

UCASE-1

```
        DATA >0038,>447C,>4444,>4400    A
        DATA >0078,>2438,>2424,>7800    B
        DATA >0038,>4440,>4044,>3800    C
        DATA >0078,>2424,>2424,>7800    D
        DATA >007C,>4078,>4040,>7C00    E
        DATA >007C,>4078,>4040,>4000    F
        DATA >003C,>405C,>4444,>3800    G
        DATA >0044,>447C,>4444,>4400    H
        DATA >0038,>1010,>1010,>3800    I
        DATA >0004,>0404,>0444,>3800    J
        DATA >0048,>5060,>5048,>4400    K
        DATA >0040,>4040,>4040,>7C00    L
        DATA >0044,>6C54,>5444,>4400    M
        DATA >0044,>6454,>4C44,>4400    N
        DATA >0038,>4444,>4444,>3800    O
        DATA >0078,>4478,>4040,>4000    P
        DATA >0038,>4444,>5448,>3400    Q
        DATA >0078,>4478,>5048,>4400    R
        DATA >003C,>4038,>0404,>7800    S
        DATA >007C,>1010,>1010,>1000    T
        DATA >0044,>4444,>4444,>3800    U
        DATA >0044,>4428,>2810,>1000    V
        DATA >0044,>4454,>5454,>2800    W
        DATA >0044,>2810,>2844,>4400    X
        DATA >0044,>2810,>1010,>1000    Y
        DATA >007C,>0810,>2040,>7C00    Z
```

PUNCT-C1

```
        DATA >0038,>2020,>2020,>3800    [
        DATA >0000,>4020,>1008,>0400    \
        DATA >0038,>0808,>0808,>3800    ]
        DATA >0010,>386C,>4400,>0000
        DATA >0000,>0000,>0000,>7C00
        DATA >0020,>1008,>0000,>0000    ~
```

LCASE-1

```
        DATA >0000,>0038,>4848,>3C00    a
        DATA >0020,>2038,>2424,>3800    b
        DATA >0000,>001C,>2020,>1C00    c
        DATA >0004,>041C,>2424,>1C00    d
        DATA >0000,>001C,>2830,>1C00    e
        DATA >000C,>1438,>1010,>1000    f
        DATA >0000,>001C,>241C,>0438    g
        DATA >0020,>2038,>2424,>2400    h
        DATA >0010,>0030,>1010,>3800    i
        DATA >0008,>0008,>0808,>4830    j
        DATA >0020,>2424,>3828,>2400    k
        DATA >0030,>1010,>1010,>3800    l
        DATA >0000,>0078,>5454,>5400    m
        DATA >0000,>0038,>2424,>2400    n
        DATA >0000,>0018,>2424,>1800    o
        DATA >0000,>0038,>2438,>2020    p
        DATA >0000,>001C,>241C,>0404    q
        DATA >0000,>0028,>3420,>2000    r
        DATA >0000,>001C,>300C,>3800    s
        DATA >0010,>1038,>1010,>0C00    t
        DATA >0000,>0024,>2424,>1C00    u
        DATA >0000,>0044,>2828,>1000    v
        DATA >0000,>0044,>5454,>2800    w
        DATA >0000,>0024,>1818,>2400    x
        DATA >0000,>0024,>241C,>0438    y
        DATA >0000,>003C,>0810,>3C00    z
```

CHARA2/S (copies for Apple font)

```
        COPY "DSK1.CTRLCHAR"
        COPY "DSK1.PUNCT-A2"
        COPY "DSK1.DIGITS-2"
        COPY "DSK1.PUNCT-B2"
        COPY "DSK1.UCASE-2"
        COPY "DSK1.PUNCT-C2"
        COPY "DSK1.LCASE-2"
        COPY "DSK1.PUNCT-D"
```

PUNCT-A2

```
        DATA >0000,>0000,>0000,>0000    (SPACE)
        DATA >0010,>1010,>1000,>1000    !
        DATA >0028,>2828,>0000,>0000    "
        DATA >0028,>7C28,>287C,>2800    #
        DATA >0038,>5430,>1854,>3800    $
        DATA >0044,>4C18,>3064,>4400    %
        DATA >0020,>5020,>5448,>3400    &
        DATA >0008,>1020,>0000,>0000    '
        DATA >0008,>1010,>1010,>0800    (
        DATA >0020,>1010,>1010,>2000    )
        DATA >0028,>107C,>1028,>0000    *
        DATA >0010,>107C,>1010,>0000    +
        DATA >0000,>0000,>0030,>1020    ,
        DATA >0000,>007C,>0000,>0000    -
        DATA >0000,>0000,>0030,>3000    .
        DATA >0004,>0810,>2040,>0000    /
```

DIGITS-2

```
        DATA >384C,>4454,>4464,>3800    0
        DATA >1030,>1010,>1010,>3800    1
        DATA >3844,>0408,>1020,>7C00    2
        DATA >3844,>0418,>0444,>3800    3
        DATA >0818,>2848,>7C08,>0800    4
        DATA >7C40,>7804,>0444,>3800    5
        DATA >1820,>4078,>4444,>3800    6
        DATA >7C04,>0810,>2020,>2000    7
        DATA >3844,>4438,>4444,>3800    8
        DATA >3844,>443C,>0408,>3000    9
```

PUNCT-B2

```
        DATA >0000,>3030,>0030,>3000    :
        DATA >0000,>3030,>0030,>1020    ;
        DATA >0000,>1830,>6030,>1800    <
        DATA >0000,>007C,>007C,>0000    =
        DATA >0000,>3018,>0C18,>3000    >
        DATA >3844,>0408,>1000,>1000    ?
        DATA >0038,>4454,>5840,>3C00    @
```

UCASE-2

```
        DATA >3844,>447C,>4444,>4400    A
        DATA >7824,>2438,>2424,>7800    B
        DATA >3844,>4040,>4044,>3800    C
        DATA >7824,>2424,>2424,>7800    D
        DATA >7C40,>4078,>4040,>7C00    E
        DATA >7C40,>4078,>4040,>4000    F
        DATA >3C40,>405C,>4444,>3800    G
        DATA >4444,>447C,>4444,>4400    H
        DATA >3810,>1010,>1010,>3800    I
        DATA >1C04,>0404,>0444,>3800    J
        DATA >4448,>5060,>5048,>4400    K
        DATA >4040,>4040,>4040,>7C00    L
        DATA >446C,>5454,>4444,>4400    M
        DATA >4464,>6454,>4C44,>4400    N
        DATA >3844,>4444,>4444,>3800    O
        DATA >7844,>4478,>4040,>4000    P
        DATA >3844,>4444,>5448,>3400    Q
        DATA >7844,>4478,>5048,>4400    R
        DATA >3844,>2010,>0844,>3800    S
        DATA >7C10,>1010,>1010,>1000    T
        DATA >4444,>4444,>4444,>3800    U
        DATA >4444,>4428,>2810,>1000    V
        DATA >4444,>4454,>5454,>2800    W
        DATA >4444,>2810,>2844,>4400    X
        DATA >4444,>2810,>1010,>1000    Y
        DATA >7C04,>0810,>2040,>7C00    Z
```

PUNCT-C2

```
        DATA >3820,>2020,>2020,>3800    [
        DATA >0000,>4020,>1008,>0400    \
        DATA >3808,>0808,>0808,>3800    ]
        DATA >0010,>386C,>4400,>0000
        DATA >0000,>0000,>0000,>7C00
        DATA >0020,>1008,>0000,>0000    ~
```

LCASE-2

```
        DATA >0000,>3804,>3C44,>3C00    a
        DATA >4040,>7844,>4444,>7800    b
        DATA >0000,>3C40,>4040,>3C00    c
        DATA >0404,>3C44,>4444,>3C00    d
        DATA >0000,>3844,>7C40,>3C00    e
        DATA >1824,>2078,>2020,>2000    f
        DATA >0000,>3844,>443C,>0438    g
        DATA >4040,>7844,>4444,>4400    h
        DATA >1000,>3010,>1010,>3800    i
        DATA >0800,>1808,>0808,>4830    j
        DATA >4040,>4448,>7048,>4400    k
        DATA >6020,>2020,>2020,>7000    l
        DATA >0000,>6C54,>5454,>4400    m
        DATA >0000,>7844,>4444,>4400    n
        DATA >0000,>3844,>4444,>3800    o
        DATA >0000,>7844,>4478,>4040    p
        DATA >0000,>3844,>443C,>0404    q
        DATA >0000,>5C60,>4040,>4000    r
        DATA >0000,>3C40,>3804,>7800    s
        DATA >2020,>7820,>2024,>1800    t
        DATA >0000,>4444,>444C,>3400    u
        DATA >0000,>4444,>4428,>1000    v
        DATA >0000,>4444,>5454,>6C00    w
        DATA >0000,>4428,>1028,>4400    x
        DATA >0000,>4444,>443C,>0438    y
        DATA >0000,>7C08,>1020,>7C00    z
```

The TI Forum continued from page 124

GOTO. I hope you can follow this.

[3] We could have used a function similar to CALL KEY (O,E,F) as in line 270. But, by using "getchar()" we accomplish the same thing in one line. Getchar waits for a keypress automatically without testing for "status."

[4] FOR-NEXT loops in c99 are three parts. Just as

```
240 FOR D = 2 TO 14
250 CALL COLOR(D,D,C)
260 NEXT D
```

accomplishes three things (set D = 2, than CALL COLOR (D,D,C), then increment D by one, then loop), the FOR loop in c99 does it all on one line. We say

```
for(d = 1;d< = 14;d + +);
```

d is set to a, then tested to see if it is less than or equal to 14. The color(d,d,c) is executed as log as d< = 14. As each color() function is executed, d is incremented by one by the "d + +" statement. All things are done with one statement. Also remember that there are multiple commands after a FOR statement in c99, they must be set off between a pair of braces. If a single statement, as we have here, they can be used without the braces.

I hope this aids you in understanding c99. If you have questions, just ask. And if you want a reply, send a self-addressed stamped envelope.

Jonathan's News

BCS Offers TI-Writer Tips & Tricks

Joyce Corker and J. Peter Hoddie of the Boston Computer Society are offering an excellent TI-Writer Tips and Tricks guide for $5.00 plus $1.00 shipping.

The guide is a helpful refresher for those who are familiar with TI-Writer but haven't looked at the manual recently, and even contains a few tips not found in the manual.

For more information or to order the guide, contact the Boston Computer Society, TI-99/4A User Group, One Center Plaza, Boston, MA 02108.

Pre-Scan It! Another Great Programmer's Aid

Also from Peter Hoddie of the BCS and distributed through Asgard Software is Pre-Scan It!, an Extended BASIC utility designed to cut down on the initial XBASIC program starting time through use of the prescan commands.

TI Extended BASIC version 110 (virtually all TI XBASIC modules now in use) includes prescan commands for use by programmers. The idea is to be able to toggle the initial prescanning of a program, which is what takes up so much time between the issuing of the RUN command and the actual execution of the program.

"!@P−" will turn off prescan, "!@P+" will turn it

continued on page 268

The Inner Limits continued from page 123

and/or Chara-AP files. Then rename one of them (the one you want to use) to Chara1, which TI-Writer will be able to load. Put this file on the same disk as the TI-Writer program files are on. TI-Writer will automatically load it.

In the file listings, the filename precedes the file in bold print. Type in that file and save it under the suggested name. **Do not enter the name into the file itself!** Purge memory and start on the next one. Given that most people do make typographical errors, using small files like this makes it easy to find errors, which will be evident because the characters will not look right.

That's it for this month. If you have any questions on these fonts, please write me care of *Computer Shopper*. I'd appreciate letters from people that have developed foreign characters for both of these fonts. THE INNER LIMITS. ●

Computer Book List
continued from page 129

and tricks. Contents include: Application development in assembler, structured programming techniques, real time programming, 8087/8088 co-processing, and recovery of lost data. A good title for the developer.

***Amiga ROM Kernel Reference Manual:* Libraries and Devices**
Edited (CBM)
Addison-Wesley — $34.95
Pages: 544—Copyright: 1986

The long awaited and eagerly sought after *ROM Kernel Manual* contains a complete listing and description of the Amiga's built in read only memory (ROM) routines which support graphics, sound and animation. Assumes a basic knowledge of C and Assembly Language. Other titles in the series include *Amiga Hardware Reference Manual* [$24.95] and *Amiga ROM Kernel Reference Manual:* exec. [$24.95] Also available are *Amiga Programmers Handbook* [$24.95], *Amiga Programming Guide* (Que) [$18.95]. *COMPUTE!'s Amiga DOS Reference Guide* [$14.95] and *COMPUTE!'s Amiga Programmers Guide* [$16.95]

Complete Turbo Pascal
Duntemann
Scott Foresman & Co.–$22.95
Pages: 540 — Copyright: 1986

Written for programmers new to Pascal, this is an excellent guide to Borland's best selling Turbo Pascal Compiler. After a complete tutorial on the Pascal language, there follow detailed instructions on installing, modifying and using Turbo Pascal on both MS-DOS and CP/M based systems.

Contents include the following: Techniques for structured programming and program design, a guide to using Turbo Editor, Instructions on Turbo Pascal commands and overlays, explanations of the extended Turbo Pascal features not covered in the manual. This very readable, comprehensive guide includes dozens of examples—from the very elementary practice problems to large, complex programs. The author includes explanations of Turbo Pascal error messages as well as a detailed glossary. Other titles include *Advanced Turbo Pascal* (Osborne McGraw-Hill), [$18.95] *Turbo Pascal: A Problem Solving Approach* (Addison Wesley) [$29.95] and a recent arrival on the scene *Turbo Pascal Program Library* from Que [$19.95]

Using Turbo Prolog
Robinson McGraw-Hill
$19.95 Pages: 300
Copyright: 1986

Using Turbo Prolog allows the programmer to maximize coding skills with this new compiler from Borland. Using multiple windows to view and modify programs while they run, this title allows you to learn programming techniques using color graphics, turtle graphics and sound. Offers detailed coverage of Prolog statements, functions and operations.

* * * * * * * * * * *

InfoSource is a full service mail order bookstore specializing in computer and technical books.

Should you wish to order any of the above titles, please send your payment, together with UPS shipping of $3.00 for the first title and 75¢ for each additional title. Most titles in print can be obtained by contacting: InfoSource, PO Box 21092, Midtown Station, New York, NY 10129; (201) 595-4505.

* * * * * * * * * * *

Readers!!!

It should be apparent from the constant stream of technical and computer books on the market, that it is difficult to keep up with the numerous titles. ●

The TI Forum
continued from page 261

back on. For programs that are thoroughly debugged, there's no need to have the entire program prescanned. There are certain rules that need to be followed for the insertion of the prescan toggle commands, though, which make modifying a program that doesn't use !@P– and !@P+ very time-consuming.

If one plans to use the prescan commands in advance of writing a program, it is not that difficult to incorporate them. All variable names must be noted, and all subprogram names (i.e. any kind of CALL command) as well. The idea is to have a few lines at the beginning of a program containing each CALL command used once, and each variable used once. The prescan can then be turned off for the rest of the program. The improper use of pre-scan commands will cause a syntax error for such valid commands as CALL CLEAR.

Pre-Scan It will take an existing XBASIC program in MERGE format on a disk and rewrite it to include the various !@P– and !@P+ commands. On the programs I sampled PSI with, the initial time between RUN and actual execution was cut in half at the least, and almost completely eliminated at best.

PSI will also optionally change five selected constants to the variables "@", "[", "]", "_", and "\", which, according to the manual, "take only ⅓ as much space." Perhaps it was the type of XBASIC programs I was experimenting with, but I noticed no decrease in overall size on disk after requesting this option.

Pre-Scan It does take awhile to modify a program—about 10-15 minutes for a 30-40 sector XBASIC program. It will work with or without the 32K Memory Expansion, and also has "modules" on disk for other brands of Extended BASIC—Myarc, Mechatronics, and Millers Graphics Extended BASIC modifications. "More files will be created for other versions as necessary and distributed free of charge," says the manual.

The documentation for PSI is acceptable, although it seems to have a lot of description of the program's features without concrete examples for use and explanations of the program's messages.

At $10, PSI is an excellent value, for both the XBASIC programmer and anyone who uses Extended BASIC programs. Most commercial XBASIC programs can be handled by PSI, so its uses are many to even the non-programmer.

When combined with Oak Tree Systems' SMASH program, another fine XBASIC utility that actually compacts lines and variables. PSI can have a major niche in the programmer's toolkit.

The program, as far as I could tell, was bug-free (although the manual had some typographical errors) and quite user-friendly.

For more information on Pre-Scan It!, or to order, contact Asgard Software, PO Box 10306, Rockville, MD 20850.

J. Peter Hoddie should be congratulated for yet another original and useful program for the 99/4A. ●