

BITS, BYTES & PIXELS

LIMA 99/4A USERS GROUP



November 1994 - Volume 10, #11

THE 1994 CHICAGO/MILWAUKEE FARE reported by Charles Good

I am sure you will read about this event in Micropendium and in other newsletters, so I will be brief. Attendance looked to me to be about 100, plus exhibitors.

Mike Wright and Mark VanCaponella of CADD Electronics demonstrated a greatly enhanced version of PC99, software that emulates the 99/4A on an IBM. I reviewed PC99 in Micropendium and my main complaint was that it was too slow. What we saw at the Chicago show was FAST. Running on a 486/50 it was as fast or faster than a real 99/4A. I was impressed! It seems the speed problems have been solved. "Review Module Library" now works. This new version of PC99 includes 3 voice music using a sound card (Bruce Harrison's classical music sounds great), the "mini screen" editor that shows the TI screen along with TI memory data simultaneously, and the ability to emulate the TI version of the P System (UCSD Pascal). Speech synthesis is not emulated. For those who already own PC99, the cost to upgrade will be minimal.

Bud Mills and Mike Maksimik talked about the SCSI project. Bud and I talked at length after the show ended. The DSR software for interfacing the SCSI card to a Geneve is "just barely not done" according to Bud. It really is almost there. Code for the 99/4A is not as far along. Brad Snyder and David Needer are working on this code. Bud told the audience at his talk that his company (Bud Mills Services) is in financial difficulty because of the SCSI project. So far he has been able to meet all his financial obligations and has issued refunds to all previous purchasers of SCSI cards who have requested refunds. However if all SCSI card purchasers to date request refunds there may be problems coming up with the money. Bud was very forthright about this. He asks for the patience of the TI community. He is NOT out of business, just short of cash.

Another seminar demonstrated an inexpensive IBM keyboard interface that lets you use either the 4A's normal keyboard or an IBM keyboard that you plug into the side of the console. You cut one trace and solder one lead to install this thing. It is AVPC compatible. Contact Bud Mills Services or Western Horizon Technologies for details. This product is currently shipping. You can get one today!

Seminars about Geneve specific topics were given by Don Walden, Tim Tesch and others. I didn't attend since I don't have a Geneve. Tim Tesch won the Birdwell Memorial award partly on the basis of his assembly language TI BBS program.

As usual lots of software and hardware changed hands. I purchased four brand new (never used, still in shrink wrapped boxes) Romox game cartridges. My kids are delighted with these! Because of the marginal attendance, some of the dealers may not have made expenses. I know that business for Bud Mills and Recharged Computers cleared only modest profits after expenses..

****DONE****

POWER YOUR 99/4A SYSTEM ANYWHERE by Charles Good Lima Ohio User Group

I have written on this topic in the past. A significant reduction in the price of the needed hardware makes the subject worth discussing again.

You can operate your complete 99/4A system anywhere you have access to a 12 volt power system. This means anywhere you have access to an automobile. All you need is a device called an "Inverter", which converts 12V DC into the well regulated 60 cycle 115V modified sine wave AC current needed to run our computer. The proper sized inverter, capable of delivering 300 watts of continuous ac power is now available from DAMARK as item B-64240-397822 for \$79.99 plus \$5.50 shipping and handling.

Plug the inverter into your automobile's cigarette lighter using the inverter's built in cigarette lighter plug or connect it directly to a battery using an alligator clip adapter available almost anywhere automobile batteries are sold. Plug an ac power strip (less than \$5 at most hardware and department stores) into the grounded ac outlet of the inverter. Then plug all your computer stuff into the power strip. That's all there is to it! The DAMARK inverter will allow you to run your entire TI system almost anywhere. 300 watts is enough to power your console, PE box, and TI color monitor. There are probably enough watts left over to power a dot matrix printer as well.

Damark
7101 Minnetka Avwe. N.
P.O. Box 29900
Minneapolis MN 55429
Phone 800-729-9000

****DONE****

Reverse Video and Funnelweb Editors Vn 5.00
By Jacques GrosLouis

The use of FWB editors offers many advantages over other text editor programs. One of these advantages in the ALL CHAR mode is that you no longer need to use the Formatter program. However, transliterate commands cannot be used and all printer commands must be inserted in the text by use of control codes. These codes usually appear as short or long bars followed by small numbers or letters which are used to denote codes 0 to 31. If you replace these codes with reverse video characters they become easier to spot in a text file. In addition the codes show the keys which created them. This article will describe how to convert your Vn 5.00 editor character sets to show reverse video control codes.

The basic principle is that you want to replace the character codes for characters 0 to 31 with reverse video codes for characters 64 to 95 (being @, all the capital letters and [, \,], ^ and _). The control code characters 0 to 31 will now appear on the screen in reverse video after being called by use of the special mode 'Control U'. As a personal preference I have not redefined characters 10, 12 and 13 which are LF, FF or p/a and c/r. Funnelweb Editor Vn 5.00 comes with source code for an assembly program called CHRCDAL/S which can be used to create reverse video characters for characters 0 to 31 (00 to 1F in hexadecimal). A listing of replacement code for CHAR00 to CHAR1F is printed below. This listing was created by converting the character codes for CHAR40 to CHAR5F to reverse video. This was done by changing each group of four hexadecimal characters using the following table:

was	0 1 2 3 4 5 6 7 8 9 A B C D E F
change to	F E D C B A 9 8 7 6 5 4 3 2 1 0

For example, the code >0038 would be changed to >FFC7. As a short aside the code on my disk for CHAR4F (the letter O) is not correct. It should start with >0038 instead of >007C. My reverse 'O' looked more like a 'U' until I made the change.

With CHRCDAL/S you do not need to use the codes supplied in the source code but may instead use code which you have extracted by using the program CHARUTIL which is also on the Funnelweb disk. In effect this is what I did because I wanted to change files C1 and CHAR@1. C1 is the file used when you are not in ALLCHAR mode and CHAR@1 is used in ALLCHAR mode. The code for C1 is shorter than the code CHAR@1 as it ends at CHAR7F. The rest of the C1 file extracted by CHARUTIL should be deleted. The procedures which I followed were as follows:

- 1- Load file CHRCDAL/S into the Program Editor.
- 2- Delete all existing CHAR00 to CHARFF codes if you created your own file using CHARUTIL.
- 3- Insert file which contains C1 code created by CHARUTIL after line beginning with CHFDAT.
- 4- Change code for CHAR00 to CHAR1F to reverse video by following the listing below. A file containing this listing could be inserted and the existing code deleted.
- 5- Save the resulting file as C1/S. The same steps would be

followed for CHAR@1 and the resulting file could be saved as CHAR@1/S.

6- Assemble both source files to produce files C1/O and CHAR@1/O.

7- Run each of these programs using L & R #4 to produce files C1 and CHAR@1.

Ensure that the existing C1 and CHAR@1 files of your FWB v5 editor are replaced with your new files. You can test your work by entering any of the editors. Type Ctrl U and then enter all the characters from @ to _ . The screen should show reverse video for each character except characters J, L and M which were not converted to reverse video. In addition, you will notice that the initial cursor is a reverse video _____. This will change to a normal cursor if you change to another cursor by using Ctrl U or Ctrl O. If you do not like this initial cursor then change CHAR1E below to >7C7C,>7C7C,>7C7C,>7C7C.

The listing of the codes for CHAR00 to CHAR1F is as follows:

```

CHAR00 DATA >FFC7,>BBA3,>ABA3,>BFC7
CHAR01 DATA >FFC7,>BBBB,>B3BB,>BBBB
CHAR02 DATA >FFB7,>DBDB,>C7DB,>DBB7
CHAR03 DATA >FFC7,>BBBB,>BFBF,>BBC7
CHAR04 DATA >FFB7,>DBDB,>DBDB,>DBB7
CHAR05 DATA >FFB3,>BFBF,>B7BF,>BF83
CHAR06 DATA >FFB3,>BFBF,>B7BF,>BFBF
CHAR07 DATA >FFC3,>BFBF,>A3BB,>BBC7
CHAR08 DATA >FFBB,>BBBB,>B3BB,>BBBB
CHAR09 DATA >FFC7,>EFEF,>EFEF,>EFC7
CHAR0A DATA >4040,>7000,>1C10,>1C10
CHAR0B DATA >FFBB,>B7AF,>9FAF,>B7BB
CHAR0C DATA >0070,>5070,>4854,>1C14
CHAR0D DATA >0070,>4070,>001C,>1010
CHAR0E DATA >FFBB,>9B9B,>ABB3,>B3BB
CHAR0F DATA >FFC7,>BBBB,>BBBB,>BBB3
CHAR10 DATA >FFB7,>BBBB,>B7BF,>BFBF
CHAR11 DATA >FFC7,>BBBB,>BBAB,>B/LB
CHAR12 DATA >FFB7,>BBBB,>B7AF,>B7BB
CHAR13 DATA >FFC7,>BBBB,>C7FB,>BBC7
CHAR14 DATA >FFB3,>EFEF,>EFEF,>EFEF
CHAR15 DATA >FFBB,>BBBB,>BBBB,>BBC7
CHAR16 DATA >FFBB,>BBBB,>D7D7,>EFEF
CHAR17 DATA >FFBB,>BBBB,>ABAB,>ABD7
CHAR18 DATA >FFBB,>BBD7,>EFD7,>BBBB
CHAR19 DATA >FFBB,>BBD7,>EFEF,>EFEF
CHAR1A DATA >FFB3,>FBF7,>EFD7,>BF83
CHAR1B DATA >FFC7,>DFDF,>DFDF,>DFC7
CHAR1C DATA >FFFF,>BFDF,>EFF7,>FBFF
CHAR1D DATA >FFC7,>F7F7,>F7F7,>F7C7
CHAR1E DATA >FFFF,>EFD7,>BBFF,>FFFF
CHAR1F DATA >FFFF,>FFFF,>FFFF,>FFB3
    
```

```

*****
*   BITS, BYTES & PIXELS   *
*   Published by Lima OH   *
*   99/4A User Group      *
*   *                       *
*   Material contained herein *
*   may be copied by any user *
*   group as long as credit *
*   is given. DV80 files of *
*   most articles in BB&P can *
*   be obtained by sending a *
*   disk and return postage. *
*   *                       *
*   ADDRESS- P.O. Box 647   *
*             Venedocia Ohio *
*             45894         *
*   Internet address:      *
*   cgood@lima.ohio-state.edu *
*   *                       *
*   Published monthly except *
*   July and August        *
*   -----               *
*   GROUP OFFICERS        *
*   President-Jennifer Poling *
*             419-667-3100 *
*   Vice Pres-Jennifer     *
*             Trudgeon    *
*   Treasurer-Leonard Cummings *
*             419-738-3770 *
*   Newsletter editor and   *
*   Librarian-Charles Good  *
*             419-667-3131 *
*****
    
```

Printer Codes and Funnelweb Editors Vn 5.00
By Jacques GrosLouis

The use of the ALL CHAR mode of Funnelweb Editors Vn 5.00 permits you to print text without using the Formatter. This article is printed in this manner and uses printer codes to format the document. I use an Epson FX-80 printer and will describe these codes.

The first line of this document contains the following code:

- 1- <ESC>"E" This causes the printer to print in emphasized mode and is entered with the following key presses: CTRL U,FCTN R,CTRL U,SHIFT F. If you have converted characters 0 to 31 to reverse video then CTRL U,FCTN R,CTRL U (which represents <ESC>) will appear on the screen as a reverse '['.
- 2- <ESC>"1" This changes line spacing to 7/72 inch and is entered by keying in CTRL U,FCTN R,CTRL U, and 1.
- 3- <ESC>"U1" This turns continuous unidirectional mode on. No spaces should appear in the printer code. This code will also work correctly on my printer if the number '1' is entered as CTRL U,SHIFT A,CTRL U. This is used to keep the sides of the border straight.
- 4- <ESC>"C"<0><11> This sets the page length at 11 inches and is keyed as CTRL U,FCTN R,CTRL U,SHIFT C,CTRL U,SHIFT 2,SHIFT K,CTRL U. SHIFT 2 produces a reverse '@' and represents character '0' while SHIFT K represents "11" being the eleventh letter.
- 5- <ESC>"N"<8> This produces an eight line skip over the end of page perforation. <8> is entered as CTRL U,SHIFT H,CTRL U because 'H' is the eight letter of the alphabet.

The border around the title of this article is created in ALL CHAR mode and uses IBM fonts stored in characters 128 to 254. My FX-80 printer does not come with this feature but permits the down loading of these characters. I had to do this and perhaps this could be the subject of another article. Further code is required in line 9 of the document as follows:

- 1- <ESC>"2" This restores line spacing to 1/6 inch(default).
- 2- <ESC>"U0" This turn unidirectional mode off.

My favourite set of printer codes sets the printer to indent 8 spaces, using 1/8 line spacing, sets form length to 88 lines and skips over perforation by 15 lines. This is very useful when doing program and source code listings and for printing out documents from DM-1000 or any other program which permits DV80 printing. The code to enter these features in DM-1000 is as follows:

27 108 8 27 77 27 48 27 67 88 27 78 15 13 *

For some reason the above will not work with Vn 6.1 of DM-1000 and I had to use the following:

```
27 68 08 00 27 77 27 48 27 67 88 27 78 15 13 *
```

The same coding used as a merge file will permit you to list an extended basic program in elite mode, indented 8 characters with skip over perforations and with printing at 8 lines per inch. This short program also allows you to date your listing and include your initials. The program which I use is as follows:

```
2 OPEN #2:"PIO" :: C#=CHR$(27):: PRINT
  #2:C#&"M"&C#&"0"&C#&"C"&CHR#(0)&CHR#(11)&
  C#&"N"&CHR$(8)&C#&"1"&CHR$(10):: INPUT "DATE?
  MM/DD/YY>":D# :: PRINT #2:TAB(65); "JJG "&D# ::
  CLOSE #2 :: STOP
```

You can use this program provided line 2 and arrays using C# and D# are not used in your program. Type in and save your program as LIST1 or any other name you like. Merge this file into any program you wish to LIST. A prompt to enter a date will appear after you run the program. Press enter and then delete line 2 by pressing 2 and then ENTER. In command mode enter LIST "PIO". The full program will then print in elite print suitable for filing in a binder by punching holes on the left side of the page.

From Funnelweb you can also set up the printer as described above or in any other mode you desire by creating text files which can be printed from the DiskReview View function or by using the Print File function of the editors. You create one file to set up the printer and another to cancel the setup in the first file. To create a file which does the same as the merge file described above proceed as follows:

1- On line 1 enter CTRL U, FCTN R, CTRL U, SHIFT M, CTRL U, FCTN R, CTRL U, 0, CTRL U, FCTN R, CTRL U, SHIFT C, CTRL U, SHIFT2, SHIFT K, CTRL U, CTRL U, FCTN R, CTRL U, SHIFT N, CTRL U, SHIFT H, CTRL U, CTRL U, FCTN R, CTRL U, lowercase l, CTRL U, SHIFT J, CTRL U. This coding is long to describe but is short on the screen.

2- Save this file using a descriptive name such as XPRLIST.

A file which would cancel the mode set up above should be prepared and saved, as say, UPRLIST. In order to run either file print from editor mode by using PF or from View mode of DISKREVIEW by first pressing 'V' to view and then CTRL P to print.

A HELP screen which I have added to my HELP files in Funnelweb editors is printed out below:

Control Codes for
Epson FX-80

	On	Off
Elite Mode	#M	#P
Compressed Mode	<O>	#
One-Line Expanded	<N>	<T>
Continuous Expanded	#W1	#W0
Emphasized Mode	#E	#F
Double Strike Mode	#G	#H
Superscript Mode	#S1	#T
Subscript Mode	#S1	#T
Proportional Mode	#p1	#p0
Underline Mode	#-1	#-0
Italic Char Set	#4	#5
Line spacing 1/8	#0	
Line spacing 7/72	#1	
Line spacing 1/6	#2	
Line spacing n/72	#Achr\$(n)	
Line spacing n/216	#3chr\$(n)	
Print char 128-159	#6	#7
Unidirectional Mode	#U1	#U0
Backspace	<H>	
ESC ctrl U fctn R ctrl U	#	
Select ROM character	##<@@>	##<A@>

Since ESC cannot be printed '#' has been used to indicate where CTRL U, FCTN R, CTRL U should be entered. Other places where CTRL U should be entered are indicated by "<" and ">".

DONE

Assembler Executing . . #9
By Bob Carmany

Let's see if we can finish up the limerick program that we started last month. I think we were at the point where the main program started. Things get just a wee bit more complicated from here on out. The preliminaries were pretty straight forward and should have been relatively easy to understand but if you have been with us so far, you should be able to understand this. Oh yes, there will be a test at the end.

*Main program begins

```

START  LWPI WKSPA           Use fast workspace
        LI   RO,>01F0       Use 40 Column mode
        BLWP @VWTR         Write to VDP register
        SWPB RO
        MOVB RO,@>83D4
        LI   RO,>07F4       White on blue
        BLWP @VWTR         write to VDP register
    
```

This is just a bit of basic housekeeping - setting the fast workspace entering the 40-column mode and writing the bytes to VDP register. The 07F4 sets the screen color to white on blue (F4). Nothing really difficult so far.

NEXT PAGE

* Clear screen and present the first bit of text

```

BL   @CLEAR           Branch to clear screen routine
BL   @SCRWRT          Branch to screen write routine
DATA 82,MSG1,34       Write at screen position 82 - MSG1 - 34 chars
BL   @SCRWRT          Etc
DATA 162,MSG2,34
BL   @SCRWRT
DATA 250,MSG3,20
BL   @SCRWRT
DATA 330,MSG4,22
BL   @SCRWRT
DATA 402,MSG5,32
BL   @SCRWRT
DATA 886,MSG11,19

```

The first operation here is to branch to the clear screen routine. That will be dealt with in a later segment of code. The next matter of business is to branch to the routine to write the message lines to the screen. A starting screen position is designated then the message number to be displayed and finally the number of characters in the message. Remember, the length of the message ALWAYS includes any spaces as well. It is the total of characters and spaces between the apostrophes following TEXT.

We are now at the point where we have the first five lines displayed on the screen as well as MSG11 at the bottom of the screen. MSG11 is the 'Press any Key' message.

The screen positions are counted from the upper left corner of the screen with the uppermost position being 0.

We must somehow erase the text from the screen and display the second limerick on the screen. For this we are going to use KSCAN and a Jump instruction.

* Advance to the next screen

```

PKEY  BLWP @KSCAN      Branch to key scan routine from REF table
      MOVB @>837C,@>837C  Move key value
      JEQ  PKEY         If they are equal (no keypress) re-scan
      BL  @CLEAR        Otherwise clear screen (key pressed)

```

The messages are on the screen and we are at the point where we are awaiting a keypress. PKEY branches to the KSCAN routine to see if a key - any key - has been pressed. The value at >837C is zero initially. That is until a key is pressed. If no key is pressed, the program continues to loop through until it detects a keypress. When it does, the bytes are not equal and the program comes out of the loop and goes to the next segment of code.

* Display next screen

```

BL   @SCRWRT           Branch to screen write for next msg group
DATA 84,MSG6,18
BL   @SCRWRT
DATA 164,MSG7,32
BL   @SCRWRT
DATA 247,MSG8,27
BL   @SCRWRT
DATA 327,MSG9,15
BL   @SCRWRT
DATA 404,MSG10,32
BL   @SCRWRT
DATA 886,MSG11,19

```

This segment of code is just like the first lot. It displays the second limerick and the 'Press any Key' message on the screen and goes to the next code segment.

* Exit routine

```
EXIT  BLWP @KSCAN           Branch to key scan routine again
      MOVB @>837C,@>837C    ( See PKEY for explanantion)
      JEQ  EXIT
      BL  @CLEAR
      BLWP @>0000           To title screen
```

EXIT is almost the same as PKEY. the difference is that when a keypress is detected, the screen is cleared and the program executes a BLWP @>0000 which is the exit to the powerup routine title screen.

* Clear screen routine

```
CLEAR LI  R0,>0040           Loading values in R0 - R2
      LI  R1,>2020           Load spaces
      LI  R2,>03C0           Screen size (number of chars)
      MOVB R0,@>8C02         Move to VDP write address
      SWPB R0
      MOVB R0,@>8C02         Move to VDP write address
CLRSCR MOVB R1,@>8C00         VDP data address
      DEC R2                 Decrement counter for screen size
      JNE CLRSCR            If not equal to total screen size keep going
      RT                    Return if screen has been clearedrn
```

The routine used to clear the screen is really not as complicated as it appears. It simply clears the screen by printing a space to the screen, decrements the counter for the total number of screen positions and keeps going until the total number of spaces equals the screen size. You then have a screen full of spaces. The RT returns to the point in the program where this routine was called.

* Write to screen routine

```
SCRWRT MOV  *R11+,R0         Move the stuff to R0 - R2 from R11
      MOV  *R11+,R1
      MOV  *R11+,R2
      BLWP @VMBW            Use VMBW to write the text to the screen
      RT
```

Another simple routine. The screen position, message, and message length are moved into R0 - R2 and then written to the screen via the VMBW (remeber that one?). It then returns to the next line in the program after it was called.

SLAST END

SLAST marks the end of the program and the only thing that can follow it in A/L code is END.

That takes us to the end of this little exercize. I think we will look at something of a bit more practical application next time. Methink will be a DSR dump from Tony McGovern.

##DONE##

A LETTER TO THE EDITOR FROM BRUCE HARRISON

27 October 1994

Dear Charlie,

Just got the November issue of BB&P today. As you might expect, I read Bob Carmany's Assembler Executing with interest, and found two things there that are just plain incorrect.

The example given for use of VMBW on Page 7 is wrong. VMBW is a vector, so one cannot BL to that utility and expect it to read DATA from the line following. VMBW requires a BLWP operation, and reads data from the calling program's R0, R1, and R2. To make Bob's example work, one would need to put in the following:

```
MSG1 TEXT 'This a test message'
REF VMBW ref the vector
...
LI R0,84 location in R0
LI R1,MSG1 text pointer in R1
LI R2,19 length of text in R2
BLWP @VMBW Branch and load workspace pointer to VMBW
```

This works! I tested Bob's example exactly as he showed it, and it just locked up the computer when it was told to BL @VMBW.

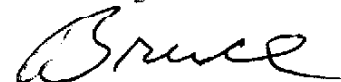
On page 8, Bob shows a "shortcut" that also won't work, because one can't CB to an immediate value. One can do that this way:

```
MOV @>8375,R1 key value to R1
SRA R1,8 right justify in R1
CI R1,'1' compare to '1' (>31)
JLT KEYPRS if less, ignore
CI R1,'9' compare to '9' (>39)
JGT KEYPRS if greater, ignore
....
```

And so on. One could put >31 in place of '1' and >39 in place of '9' above, with exactly the same results. The Assembler will translate '1' into the immediate value >0031 for you.

I've talked about this on the phone with Bob, and sent him a copy of this letter. Please publish this in BB&P so people won't get frustrated trying things that won't work.

Best Regards,



cc: Bob Carmany

DONE