THE PUG PERIPHERAL

THE MONTHLY NEWSLETTER OF THE

PITTSBURGH USERS GROUP

OCTOBER 1991

Club News by Gary Taylor

The meeting this month and next month will be held on the 1st Sunday of the month, October 6 and November 3, instead of the 2nd Sunday of the month. The Whitehall community center will be using the building on the 2nd Sunday for those two months so we were given the 1st Sunday for our meetings.

I will be having a one hour session beginning at 4:00 on DISK+AID, a really nice disk sector editing program. This used to be a commercial program but was released as fairware a few years ago. It is also the one that I am most familiar with as I use it for recovering data from trashed diskettes. My skills really became sharpened after using this program to straighten out the files that were clobbered by the Myarc HFDC. If you want to find out how to use a disk sector editor then you don't want to miss this discussion. The program is in our library and we will have extra copies on hand for the meeting. In preparation for this session I have uploaded a copy of the documentation to our BBS. It is comprised of three files on disk 2. They can be downloaded as ASCII files or by using Xmodem transfers. I will also have copies of the docs on disk at the meeting.

At 5:00 Mickey Schmitt will begin a one hour class on basic programming. This is the first class on basic programming that we have had in a while and the first one taught by Mickey at our meeting. She has become quite proficient in the last couple of years and will be able to answer any questions you have. This is the first class so you can get in on the ground floor, so to speak.

The bbs is back up and running at 2400 bps. As I mentioned at the last meeting, our 2400 bps modem failed and I installed a 300 bps modem on the bbs temporarily. We were able to get a 2400 bps modem without breaking the bank and it has been running smoothly ever since. THANKS to Chris Pratt for helping us obtain the modem.

Speaking of Chris, he recently announced that the long awaited ESD disk controller will be available soon. It is now packaged as a hard disk controller only and will sell for $275. I understand that it demonstrated at the Manners User group meeting last month. Since it only controls a hard disk, you will have to keep your floppy disk controller in your P_box.

I will also be demonstrating the Milton Bradley MBX system at the meeting. Milton Bradley introduced this system about the time that TI left the market place in 1983 and 1984. The system has built in speech capabilities as well as voice recognition. Several of the games they introduced use a microphone to control the action on the screen. I have several of the games and will show you how the speech recognition is set up and used. This is really a clever addition to the capabilities of the TI computer. One that has not been duplicated, that I know of, on other computers for the home market.

THE KIDDIE CORNER

by Sue Harper
Pittsburgh User Group

For kids of all ages - a series of articles on how to get started making your own programs.

Last month I proposed writing a program of some length and difficulty. Well, now is the time to get started doing just that, but for this month we will not actually have any program in the article - that part will be your 'homework', should you choose to join me in this project. Let's begin with section one of the program, the introduction, and how we want to lay out the section.

First, it is important to understand how the computer thinks, which is generally in a straight line. In other words, the program will logically follow along, with the lowest numbered line first, and count up from there. Therefore, the first thing to do after turning on the computer is to type in the command NUM. NUM will tell the computer that you want it to count for you, and keep track of the line numbers for you. It will start at 100, and count by tens. It is important to not number your lines 1, 2, 3, and so on, because you might forget something and then have no line number to go back and add to the program. There is, however a solution to that even, by using the command RES. RES is short for resequence, which tells the computer that you want to renumber the lines. The computer will then change the line numbers to begin with 100 and count by tens. It will even change the line numbers inside the program, such as in GOTO statements. For example:

```
  CALL CLEAR
7 PRINT "HELLO"
10 GOTO 7

RES
100 CALL CLEAR
110 PRINT "HELLO"
120 GOTO 110
```

You can use this if you make an error.

Next, use REM statements in your program. These statements are remarks in the program for the programmer or debugger to know what is supposed to be happening.

```
NUM
100 CALL CLEAR
110 REM THIS PROGRAM IS WRITTEN IN BASIC
120 REM SPEACH MODULE AND TEII REQUIRED
130 PRINT TAB(20):"TI WORLD"::::::::::::::
140 FOR WAIT=1 TO 250
150 NEXT WAIT
160 PRINT"PRESS ANY KEY TO BEGIN"
170 CALL KEY(0,K,S)
180 IF S=0 THEN 170
190 REM THIS SECTION GIVES THE CHOICES
200 CALL CLEAR
```

Well, I hope you get the idea, and I have already given you a head start on your program. Next, use PRINT statements to list the options, and IF-THEN-ELSE statements to tell the program where to go. Remember: if you tell the program what line number to go to, it will, even if it is out of sequence. If you do not, the program will automatically look for the next highest line number.

Happy programming! See you next month.

FORMATTER TIP

Reprinted from Mid South 99

A real timesaver for people who use the TIW Formatter in FWeb. You can do a disk directory while in the Editor and mark a file so that you don't have to type in the DSKx.FILENAME. This is a big help if you can't remember the filename. If you do a disk directory while in the Formatter, you can't mark the file, so if you want to mark a file for the mailbox, you exit the Formatter, enter the Editor, do a disk directory, mark the file, and exit the Editor, reenter the Formatter. This is very clumsy and slow if you are not using a RAMDISK.
John Briscoe's trick has you do a disk directory (Fctn)7 while still in the Formatter. Arrow down to the file you want. Press the space bar, which places an invisible mark on the file. Press <Ctrl> = to return to the Formatter, then press <Fctn> D (right arrow) to place the new file name in the Formatter mail box. Saves the see saw time for repeatedly loading the formatter and editor just to mark files. This is super for people who are intimidated by long filenames and can't remember, was it DOCS or —DOCS— or —READ-ME— or README. If you give up and use single character filenames, then a year from now, when you are reviewing your disks, you won't have a clue as to what file X is for.

TIPS FROM THE TIGERCUB

#58.1

Tigercub Software
156 Collingwood Ave.
Columbus OH 43213

I am still offering over 120 original programs at $1 each, or on collection disks at $5 each. The five Tips From The Tigercub disks are reduced to $5 and the three Nuts & Bolts disks are now just $10 each.

My catalog is available for $1, deductable from your first order (specify TIGERCUB catalog).

****

TI-PD LIBRARY

I have selected public domain programs, by category, to fill over 300 disks, as full as possible if I had enough programs of the category, with all the Basic-only programs converted to XBasic, with an E/A loader provided for assembly programs if possible, instructions added and any obvious bugs corrected, and with an autoloader by full program name on each disk. These are available as a copying service for just $1.50 postpaid in U.S. and Canada. No fairware will be offered without the author's permission. Send SASE for list or $1, refundable for 11-page catalog listing all titles and authors. Be sure to specify TI-PD catalog.

****

In Tips #55 I published a CHARSUB routine to convert character patterns into assembly source code, and in Tips #55 and #56 I published several routines to manipulate hex codes into new character sets. Those patterns looked fine on my old TV, but when I demo'd them on a high-resolution monitor I could see too many missing pixels.

So I wrote this CHARFIX program which, when MERGEd into a program and CALLed after any character redefintion is completed, will permit any normal or reidentified character to be viewed on screen and edited and will then write the hex codes of any range of printable characters into an assembly source file which can be assembled, loaded and linked to instantly change character sets.

This routine also reidentifies the common punctuation into the same character sets as the letters, as described in Tips #55. If you do not want this feature, delete lines 29001-29003.

```
29000 SUB CHARFIX
29001 DATA 32,33,34,44,46
29002 RESTORE 29001 :: FOR J
=1 TO 5 :: READ CH :: CALL C
HARPAT(CH,CH$):: CALL CHAR(J
+90,CH$):: CALL CHAR(J+122,C
H$):: NEXT J
29003 CALL CHARPAT(63,CH$)::
 CALL CHAR(64,CH$):: CALL CH
AR(96,CH$)
29004 DISPLAY AT(1,1)ERASE A
LL:"1 2 3 4 5 6 7 8 9 0 : ;"
:" ":"@ A B C D E F G H I J
K L M":" ":"N O P Q R S T U
V W X Y Z [":" ":"\ ] ^ _ a
b c d e f g h i j"
29005 DISPLAY AT(9,1):"k l m
 n o p q r s t u v w x":" ":
"y z { | } ~"
29006 CALL CHAR(128,"FF"&RPT
$("81",6)&RPT$("FF",9)&"FFFF
"&RPT$("C3",4)&"FFFF"):: CAL
L COLOR(13,2,16)
29007 CALL CHARVIEW
29008 SUBEND
29009 SUB CHARVIEW
29010 DISPLAY AT(13,14):"CTR
L V TO VIEW" :: DISPLAY AT(1
4,14):" " :: DISPLAY AT(15,1
4):"CTRL E TO EDIT" :: DISPL
AY AT(17,14):"CTRL S TO SAVE
"
29011 DISPLAY AT(19,14):" "
:: DISPLAY AT(20,14):" "
29012 CALL KEY(0,@,S):: IF S
=0 THEN 29012 ELSE IF @=150
THEN 29015 ELSE IF @=133 THE
N 29014 ELSE IF @=147 THEN 2
9013 ELSE 29012
29013 CALL DELSPRITE(#1):: C
ALL CHARSUB(HX$()):: DISPLAY
 BEEP :: STOP
29014 CALL EDIT(K):: GOTO 29
010
29015 DISPLAY AT(24,1)BEEP:"
"
29016 DISPLAY AT(24,1):"PRES
S A KEY" :: CALL KEY(0,K,S):
: IF S<1 OR K<32 OR K>143 TH
EN 29016
29017 DISPLAY AT(24,1):"" ::
 CALL CHARPAT(K,CH$)
29018 R=13 :: FOR J=1 TO 15
STEP 2
29019 H$=SEG$(CH$,J,1):: CAL
L HEX_BIN(H$,B$)
29020 H$=SEG$(CH$,J+1,1):: C
ALL HEX_BIN(H$,BB$):: FOR L=
1 TO 8 :: C$=C$&CHR$(ASC(SEG
$(B$&BB$,L,1))+80):: NEXT L
29021 DISPLAY AT(R,1):C$;::
DISPLAY AT(R,10):SEG$(CH$,J,
2);:: R=R+1 :: C$="" :: NEXT
 J :: DISPLAY AT(22,1):CH$;:
: GOTO 29012
29022 SUBEND
29023 SUB HEX_BIN(H$,B$):: H
X$="0123456789ABCDEF" :: BN$
="0000X0001X0010X0011X0100X0
101X0110X0111X1000X1001X1010
X1011X1100X1101X1110X1111"
29024 FOR J=LEN(H$)TO 1 STEP
 -1 :: X$=SEG$(H$,J,1)
29025 X=POS(HX$,X$,1)-1 :: T
$=SEG$(BN$,X*5+1,4)&T$ :: NE
XT J :: B$=T$ :: T$="" :: SU
BEND
29026 SUB CHARSUB(HX$())
29027 DISPLAY AT(12,1)ERASE
ALL:"Source code filename?":
"DSK" :: ACCEPT AT(13,4)SIZE
(12)BEEP:F$ :: OPEN #1:"DSK"
&F$,OUTPUT
29028 DISPLAY AT(15,1):"LINK
ABLE program name?" :: ACCEP
T AT(16,1)SIZE(6):P$
29029 DISPLAY AT(18,1):"Rede
fine characters from    ASCI
I     to ASCII"
29030 ACCEPT AT(19,7)VALIDAT
E(DIGIT)SIZE(3):F
29031 ACCEPT AT(19,21)VALIDA
TE(DIGIT)SIZE(3):T
29032 PRINT #1:TAB(8);"DEF";
TAB(13);P$ :: PRINT #1:"VMBW
   EQU  >2024" :: PRINT #1:"
STATUS EQU  >837C"
29033 NB=(T-F)*8 :: CALL DEC
_HEX(NB,H$):: A=768+F*8 :: C
ALL DEC_HEX(A,A$)
29034 FOR CH=F TO T :: IF CH
<144 THEN CALL CHARPAT(CH,CH
$)ELSE CH$=HX$(CH)
29035 IF FLAG=0 THEN PRINT #
1:"FONT";:: FLAG=1
29036 FOR J=1 TO 13 STEP 4 :
: M$=M$&">"&SEG$(CH$,J,4)&",
" :: NEXT J :: M$=SEG$(M$,1,
23)&"  *"&CHR$(CH)
29037 PRINT #1:TAB(8);"DATA
"&M$ :: M$="" :: NEXT CH
29038 PRINT #1:P$;TAB(8);"LI
   R1,FONT" :: PRINT #1:TAB(
8);"LI   R0,>"&A$ :: PRINT #
1:TAB(8);"LI   R2,>"&H$
29039 PRINT #1:TAB(8);"BLWP
@VMBW":TAB(8);"CLR  @STATUS"
:TAB(8);"RT":TAB(8);"END" ::
 CLOSE #1
29040 SUBEND
29041 SUB DEC_HEX(D,H$)
29042 X$="0123456789ABCDEF"
:: A=D+65536*(D>32767)
29043 H$=SEG$(X$,(INT(A/4096
)AND 15)+1,1)&SEG$(X$,(INT(A
/256)AND 15)+1,1)&SEG$(X$,(I
NT(A/16)AND 15)+1,1)&SEG$(X$
,(A AND 15)+1,1):: SUBEND
29044 SUB EDIT(CH)
29045 DISPLAY AT(13,14):"1 T
O TOGGLE" :: DISPLAY AT(14,1
5):"CURSOR" :: DISPLAY AT(15
,14):"E S D X TO MOVE" :: DI
SPLAY AT(17,14):"CTRL A TO A
BORT"
29046 DISPLAY AT(19,14):"CTR
L R TO" :: DISPLAY AT(20,15)
:"REIDENTIFY"
29047 R=13 :: C=3 :: X=128 :
: CALL SPRITE(#1,130,11,R*8-
7,C*8-7):: X$=CHR$(129)&CHR$
(146)
29048 CALL KEY(0,K,S):: IF S
<1 THEN 29048 ELSE ON POS("1
EeSsDdXx"&X$,CHR$(K),1)+1 GO
TO 29048,29049,29050,29050,2
9051,29051,29052,29052,29053
,29053,29055,29056
29049 X=X+1+(X=129)*2 :: GOT
O 29054
29050 R=R-1-(R=13):: GOTO 29
054
29051 C=C-1-(C=3):: GOTO 290
54
```

```
29052 C=C+1+(C=10):: GOTO 29
054
29053 R=R+1+(R=20)
29054 CALL LOCATE(#1,R*8-7,C
*8-7):: CALL HCHAR(R,C,X)::
GOTO 29048
29055 CALL DELSPRITE(#1):: S
UBEXIT
29056 FOR R=13 TO 20 :: FOR
C=3 TO 10 :: CALL GCHAR(R,C,
GH):: CALL LOCATE(#1,R*8-7,C
*8-7):: B$=B$&CHR$(GH-80)::
NEXT C
29057 CALL BIN_HEX(B$,H$)::
DISPLAY AT(R,10):H$;:: B$=""
 :: HEX$=HEX$&H$ :: NEXT R :
: DISPLAY AT(22,1):HEX$;:: C
ALL CHAR(CH,HEX$):: HEX$=""
29058 CALL DELSPRITE(#1):: F
OR R=13 TO 20 :: DISPLAY AT(
R,14):"" :: NEXT R :: SUBEND
29059 SUB BIN_HEX(B$,H$):: H
X$="0123456789ABCDEF" :: BN$
="0000X0001X0010X0011X0100X0
101X0110X0111X1000X1001X1010
X1011X1100X1101X1110X1111"
29060 L=LEN(B$):: IF L/4<>IN
T(L/4)THEN B$="0"&B$ :: GOTO
 29060
29061 FOR J=L-3 TO 1 STEP -4
 :: X$=SEG$(B$,J,4)
29062 X=(POS(BN$,X$,1)-1)/5
:: T$=SEG$(HX$,X+1,1)&T$ ::
NEXT J :: H$=T$ :: T$="" ::
SUBEND
```

I think that programs, at least non-commercial ones, should be open for anyone to modify for their own use. For that reason, I would not normally publish the following routine. However, I recently received a large number of programs, originally in the IUG library, and found that the author's name had been erased from the title screen or REM of every one of them. I know, because I already had many of the original versions, including some that I wrote myself.

Now, that is inexcusable. If a programmer is willing to share his work, he does deserve credit for it. And if people are going to play that dirty, maybe there is good reason for protecting programs.

So here is how to do it. Ken Woodcock wrote this ingenious routine and published it in the Tidewater newsletter. I have modified it so that it can be deleted after it has done its work. It is to be MERGEd into any XBasic program(32k required) and RUN, and will change the line length byte of each line to zero, so that the program cannot be LISTed, although it can be loaded and run.

```
1 CALL INIT :: CALL PEEK(-31
952,A,B,C,D):: SL=C*256+D-65
539 :: EL=A*256+B-65536 :: F
OR X=SL TO EL STEP -4
2 CALL PEEK(X,E,F,G,H):: ADD
=G*256+H-65536 :: J=J+1 :: I
F J<4 THEN 3 :: CALL LOAD(AD
D-1,0)
3 NEXT X :: STOP :: !@P-
```

Save that as FIX in MERGE format. Merge it into any program (RESequence first if it has line numbers less than 4) and RUN. Then type 1, FCTN X and FCTN 3 to delete line 1. Delete lines 2 and 3 in the same way. Then SAVE. Now try LISTing it and watch the fireworks.

Ken wrote an even more ingenious UNFIX routine to unprotect the program, but I'm not passing that on!

Now, suppose you have a party game program that you don't want the kids playing with. So, RESequence it to some odd number, such as RES 797. Put in a line just before that 796 STOP . Then merge in FIX, run it, and delete those first 3 lines.

I hope you remember what line number you resequenced it to start from, because now you can only run it by RUN 797 !

In Tips #57 I reported the discovery that printing to the disk from the TI- Writer Formatter, with the C option, really converted the carriage returns to trailing blank ASCII 32's, and I published a routine to strip them. I have found an easier way. First PF and C DSK... to convert the CRs to blanks. LF DSK... and SF DSK... to strip out those blanks, but that leaves the pestiferous tab line, so LF DSK... and PF DSK... again!

The first few disks of Tips #58 that I sent out had a poor version of this program. This is the corrected version. First key this in -

```
1 DISPLAY AT(12,1)ERASE ALL:
"SKIP INSTRUCTIONS? Y" :: AC
CEPT AT(12,20)SIZE(-1)VALIDA
TE("YNyn"):@@$ :: IF @@$="Y"
 OR @@$="y" THEN 8
2 DISPLAY AT(24,5)ERASE ALL:
"PRESS ANY KEY"
3 RESTORE 30721
4 REM
5 FOR J@=1 TO T@ :: READ @$
:: DISPLAY AT(J@,1):@$:" "
6 CALL KEY(0,K@,S@):: IF S@=
0 THEN 6
7 NEXT J@
8 DATA 0
9 RESTORE 8 :: READ N
10 REM
```

Save it by -
SAVE DSK1.D/MERGE,MERGE
Then key this in -

```
100 OPEN #1:"DSK1.D/MERGE",V
ARIABLE 163,INPUT :: OPEN #2
:"DSK1.D/MERGE2",VARIABLE 16
3,OUTPUT :: L=129 :: FOR J=1
 TO 10
110 LINPUT #1:M$ :: PRINT #2
:CHR$(0)&CHR$(L+J)&CHR$(156)
&CHR$(253)&CHR$(200)&CHR$(1)
&"1"&CHR$(181)&CHR$(199)&CHR
$(LEN(M$))&M$&CHR$(0):: NEXT
 J
120 CLOSE #1 :: PRINT #2:CHR
$(255)&CHR$(255):: CLOSE #2
```

Run it to convert D/MERGE into a merge format file D/MERGE2 on DSK1. Then key this in. Don't change line numbers.

```
100 CALL CLEAR :: OPEN #1:"D
SK1.@DATA",VARIABLE 163,OUTP
UT :: DEF L$(X)=CHR$(120)&CH
R$(X)
105 PRINT #1:L$(X)&CHR$(161)
&CHR$(200)&CHR$(6)&"@DUMMY"&
CHR$(0)
110 L=L+1 :: X=X+1 :: ACCEPT
 AT(L,0):M$ :: IF L=24 THEN
CALL CLEAR :: L=0
120 IF M$<>"END" AND M$<>"en
d" THEN PRINT #1:L$(X)&CHR$(
147)&CHR$(199)&CHR$(LEN(M$))
&M$&CHR$(0):: GOTO 110
130 REM
140 PRINT #1:CHR$(0)&CHR$(4)
&"T@"&CHR$(190)&CHR$(200)&CH
R$(LEN(STR$(X-1)))&STR$(X-1)
&CHR$(0)
141 PRINT #1:L$(X)&CHR$(168)
&CHR$(0)
150 PRINT #1:CHR$(255)&CHR$(
255):: CLOSE #1
```

Enter MERGE DSK1.D/MERGE2 to merge in that file. SAVE the program as DATAWRITER. Then RUN it and try it out by using it to write itself some instructions. Answer the prompts with -

DATAWRITER V1.2
by Jim Peterson

To be used to add instructions to programs.

Type the instructions and format them, centered or hyphenated or right-adjusted just as you want them to appear on screen, and enter each line. They will be written to a D/V163 file named @DATA. When finished, enter END.

Then enter NEW, then MERGE DSK1.@DATA, and RUN to see if everything is OK. If so, load the program needing instructions, make sure its lowest line number is more than 10 and the highest is less than 30721, and enter MERGE DSK1.@DATA.

And enter END, then OLD DSK1.DATAWRITER, then MERGE DSK1.@DATA.

MODEMS, Part 2
by Al Kinney

REPRINTED FROM PUNN

In last month's article, we began by reviewing the structure of data as it applies to telecommunications and modems. This month, then, we'll add some more supporting information.

In addition to understanding the data you want to transmit, you should understand the basic terms that relate to data transfer. This section will explain all those technical words that pop up when the subject is data communications - words like *parity*, *asynchronous*, *bps*, *baud*, *full duplex*, *half duplex*, *XON/XOFF* and *flow control*.

What is Parallel or Serial Transmission?

Now that you understand that data is made of bits and bytes, it will be easy for you to understand how the data is sent from one computer to another. Basically, the computer sends out one kind of electrical signal for a "1" bit and a different signal for a "0" bit.

To understand parallel transmission, think about a large number of people trying to get into a movie. If eight entrances are available at the same time, eight people can get into the movie at the same time. If only one turnstile is available, only one person can enter at a time. When eight data lines move data at the same time, this is said to be parallel operation, while if only a single line is used, it is said to be serial operation.

There are, of course, advantages and disadvantages to using one over the other, but for modem use, it's only practical to use serial, since parallel requires 8 wires to transmit and receive the data, while serial only requires a single wire. Another major advantage of serial transmission is the ability to use standard telephone lines, which isn't possible using parallel connections.

Data Speed - Baud and bps

Terminology which relates to data speed is frequently misunderstood, even by some so-called "experts." Serial data speed is expressed as the number of bits transmitted per second (*bps*). This is often referred to as the *baud* rate, although baud and bps are not necessarily the same. When data speed is given in *bps*, the actual number of bits transmitted per second is specified. When data speed is given in baud, however, this is not true. You already know that data is sent as electrical signals and that the electrical signal for a "1" bit is different from the signal for a "0" bit. A *baud* is a unit of signaling speed that measures the number of electrical signals sent on the line during one second, much like a rain gauge measures inches of rainfall per hour. The baud is named for Frenchman J.M.E. Baudot, who developed the codes used for telegraph systems.

MODEMS, Part 2

If only one bit is transmitted per signal, the bit rate (bps) is the same as the baud rate. Otherwise the two are not equivalent. Typically, more than one bit is sent in one signal, and more than one signal is usually sent in one second. For example, a modem which sends data at 2400 baud can send data at 2400 bps if one bit is sent per signal, at 4800 bps if two bits are sent per signal, or 9600 bps if 4 bits are sent per signal. Special codes can be used to send more than one bit in a signal. The following data speeds are commonly used in data communications: 300, 1200, 2400, 4800, 9600 and 19200 bits per second. Be careful not to confuse baud and bps: *baud* is gradually being replaced by the term *bits per second* because bps has the same meaning all the time, whereas baud does not.

Remember two things about data rates:

- Two pieces of equipment must transmit at the same data rate in order to communicate.
- A common mistake in data communications is to use too high a data rate. For instance, while the port of the TI-99/4A will support 9600 bps operation I haven't found a comm program, which will manage a rate that high, with any consistency.

Synchronous or Asynchronous Transmission

Anything that is *synchronous* is done regularly, and in telecommunications, according to a timing signal. Data bytes are send down the line, one right after another, with no delays in between. Here is a picture of what data looks like when it is being transmitted synchronously:

>... BYTE 3 BYTE 2 BYTE 1 ...>

If a communications line is *asynchronous*, a delay may occur between bytes, but a delay will *not necessarily* occur between every pair of bytes. Here is what data looks like when it is being transmitted asynchronously:

>.. BYTE 4 BYTE 3 .. BYTE 2 .. BYTE 1 ..>

Asynchronous communications is simpler and less expensive than synchronous, because synchronous communications requires special hardware which is often quite costly. Asynchronous communications is most often used by micro and minicomputers, and we'll only consider asynchronous communications in these articles.

Next month, we'll continue discussing Communications Parameters...DUPLEX, Full or Half?

PROGRAMS THAT WRITE PROGRAMS PART 6

by Jim Peterson

The first five parts of this series were written long ago, but since then I have found a new method to write programs that really do write programs. I must give Karl Romstedt credit for this idea. To illustrate this technique, I will use a program which writes an auto-loader to display a diskfull of programs by their complete name rather than the abbreviated filename. This is the LOAD program which I put on all my TI-PD disks. First, we key in the part which will always be a part of the LOAD program. Do not change the line numbers because there is a reason for them, and leave that REM in line 11 because something else will be plugged in there later.

```
10 CALL CLEAR :: DIM M$(127)
:: CALL SCREEN(5):: FOR S=0
TO 14 :: CALL COLOR(S,2,8)::
 NEXT S :: CALL PEEK(8198,A)
:: IF A<>170 THEN CALL INIT
11 REM
12 ON WARNING NEXT
13 X=X+1 :: READ M$(X):: IF
M$(X)<>"END" THEN 13
14 R=3 :: FOR J=1 TO X-1 ::
READ X$ :: DISPLAY AT(R,1):S
TR$(J);TAB(4);X$ :: R=R+1 ::
 IF R<23 THEN 17
15 DISPLAY AT(24,1):"Choice?
 or 0 to continue 0" :: ACCE
PT AT(24,26)VALIDATE(DIGIT)S
IZE(-3):N :: IF N>X-1 THEN 1
5
16 IF N<>0 THEN 19 :: R=3
17 NEXT J
18 DISPLAY AT(24,1):"Choice?
" :: ACCEPT AT(24,9)VALIDATE
(DIGIT):N :: IF N=0 OR N>X-1
 THEN 18
19 CALL CHARSET :: CALL CLEA
R :: CALL SCREEN(8):: CALL P
EEK(-31952,A,B):: CALL PEEK(
A6+B-65534,A,B):: C=A6
+B-65534 :: A$="DSK1."M$(N)
:: CALL LOAD(C,LEN(A$))
20 FOR J=1 TO LEN(A$):: CALL
 LOAD(C+J,ASC(SEG$(A$,J,1)))
:: NEXT J :: CALL LOAD(C+J,0
):: GOTO 10000
10000 RUN "DSK1.1234567890"
```

Now, save that "source code" by SAVE DSK1.CAT/S,MERGE. Then key in this "assembler" which will convert the "source code" into an "object code."

```
100 OPEN #1:"DSK1.CAT/S",VAR
IABLE 163,INPUT
110 OPEN #2:"DSK1.CAT/O",VAR
IABLE 163,OUTPUT
120 FOR J=10 TO 21 :: LINPUT
 #1:M$ :: PRINT #2:CHR$(0)C
HR$(J)CHR$(156)CHR$(253)C
HR$(200)CHR$(1)"2"CHR$(18
1)CHR$(199)CHR$(LEN(M$))M
$CHR$(0):: NEXT J
130 PRINT #2:CHR$(255)CHR$(
255):: CLOSE #1 :: CLOSE #2
```

Note what this routine does. It reads in each line of the tokenized CAT/S and prints it back out to CAT/O preceded by line numbers 10 to 21 in tokenized two-byte format followed by the tokens for PRINT #2, the tokens for a quoted string followed by the CAT/S record and the CHR$(0) end-of-line indicator. Then it imprints the double-255 end-of-file indicator and closes the files.

Now key in the CATWRITER program.

```
1 CALL CLEAR :: CALL TITLE(1
6,"CATWRITER"):: CALL CHAR(1
27,"3C4299A1A199423C"):: DIS
PLAY AT(2,10):"Version 1.4":
::TAB(8): Tigercub Softwar
e"
2 DISPLAY AT(15,1):"For free
":"distribution":"but no pri
ce or":"copying fee":"to be
charged." :: FOR D=1 TO 500
:: NEXT D :: CALL DELSPRITE(
ALL)
3 DISPLAY AT(2,3)ERASE ALL:"
TIGERCUB CATWRITER V.1.4"::
" Will read a disk directory
,":"request an actual progra
m":"name for each program-ty
pe"
4 DISPLAY AT(7,1):"filename,
 and create a merg-":"able Q
uickloader which dis-":"play
s full program names and":"r
uns a selected program."
5 DISPLAY AT(12,1):" Place d
isk to be cataloged":"in dri
ve 1 and press any key" :: C
ALL KEY(0,K,S):: IF S=0 THEN
 5
9 OPEN #2:"DSK1.CAT",VARIABL
E 163,OUTPUT
100 OPEN #1:"DSK1.",INPUT ,R
```

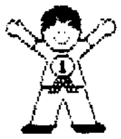
```
ELATIVE,INTERNAL :: INPUT #1
:N$,A,J,K :: LN=1000 :: FN=1
100
110 DISPLAY AT(12,1):"Disk n
ame?":;:N$ :: ACCEPT AT(14,1
)SIZE(-28):N$ :: LX$=STR$(14
-LEN(N$)/2):: LXLEN=LEN(LX$)
120 PR$=CHR$(0)CHR$(11)CHR
$(162)CHR$(240)CHR$(183)C
HR$(200)CHR$(1)"1"CHR$(17
9)CHR$(200)CHR$(LXLEN)LX$
130 PR$=PR$CHR$(182)CHR$(1
81)CHR$(199)CHR$(LEN(N$))
N$CHR$(0)
140 PRINT #2:PR$
145 DISPLAY AT(23,1):"To omi
t a file, press Enter"
150 X=X+1 :: INPUT #1:P$,A,J
,B :: IF LEN(P$)=0 THEN 190
:: IF ABS(A)=5 OR ABS(A)=4 A
ND B=254 THEN 160 ELSE X=X-1
 :: GOTO 150
160 DISPLAY AT(12,1):P$:"
 PROGRAM NAME?" :: ACCEPT AT
(14,1)SIZE(25):F$ :: IF F$="
" THEN X=X-1 :: GOTO 150
170 PRINT #2:CHR$(INT(FN/256
))CHR$(FN-256*INT(FN/256))
CHR$(147)CHR$(200)CHR$(LEN
(F$))F$CHR$(0):: FN=FN+1
180 M$=M$CHR$(200)CHR$(LEN
(P$))P$CHR$(179):: IF X<11
 THEN 150
190 IF M$="" THEN 210
200 PRINT #2:CHR$(INT(LN/256
))CHR$(LN-256*INT(LN/256))
CHR$(147)SEG$(M$,1,LEN(M$)-
1)CHR$(0):: LN=LN+1 :: M$="
" :: X=0 :: IF LEN(P$)<>0 TH
EN 150
210 PRINT #2:CHR$(INT(LN/256
))CHR$(LN-256*INT(LN/256))
CHR$(147)CHR$(200)CHR$(3)
"END"CHR$(0)
220 PRINT #2:CHR$(255)CHR$(
255):: CLOSE #1 :: CLOSE #2
230 DISPLAY AT(8,1)ERASE ALL
:"Enter -":;:"  NEW":;:"  ME
RGE DSK1.CAT":;:"  DELETE ""
DSK1.CAT""":;:"  SAVE DSK1.L
OAD"
240 SUB TITLE(S,T$)
250 CALL SCREEN(S):: L=LEN(T
$):: CALL MAGNIFY(2)
260 FOR J=1 TO L :: CALL SPR
ITE(#J,ASC(SEG$(T$,J,1)),J+1
-(J+1=S)+(J+1=S+13)+(J>14)*1
3,J*(170/L),10+J*(200/L))::
NEXT J
270 SUBEND
```

Next, enter MERGE DSK1.CAT/O and that "object code" will pop into place right after line 9. If you list it, it will look like a blown file, because most of the token codes are unprintable, but don't worry. Save the program as CAT-WRITER.

When you run the program, it will open an output MERGE format file called CAT and write those merged lines from CAT/O in MERGE format. Then it will open the disk you are cataloging, read the directory sector, and ask you for a disk name with the existing diskname as default. You can select any disk name you want to title the menu screen, up to 28 characters long. Line 110 computes the position to center the title, and lines 120-140 write to the CAT file a tokenized line 11 (overwriting that REM line) to display your title at the top of the screen. Line 150 reads each filename from the disk directory, skipping over anything that is not a program (no one yet has been able to tell me how to distinguish an assembly image "program"!). For each filename, it will ask you for a complete program name. If you don't want a program on the menu (such as an XB program that is run from another program, or an image file), just press Enter. Otherwise the program name you select will be printed as DATA by line 170, in tokenized format in lines starting with 1100 (note the FN=1100 in line 100) and incremented by 1. Lines 180-200 assemble the filenames into DATA lines of up to ten names, and tokenize them in lines beginning with 1000. When the last filename has been read, line 210 prints one last DATA item "END" to signal line 13 to stop reading, and then prints the double-255 end-of-file. Then you are given instructions to clear memory with NEW, merge in the CAT file, delete it because you don't need it any more, and save it back as LOAD. When you list the LOAD program, you will find the original CAT/S restored in lines 10-19 and 1000, the line to display the title in line 11, the filenames in DATA lines starting with 1000 and the program names in DATA lines starting at 1100. When you run the program, it will display the disk name, and read the file names into an array. Then it will display the program names, numbered, on as many screens as necessary, and ask you to select a program by number. The corresponding filename by number is selected from the array, and lines 19-20 rewrite line 10000 to RUN that filename. List the LOAD program after you have used it to load something, and you will see that it has changed.

That algorithm in lines 19-20 was published in one of the earliest 99'ER magazines, in a letter by A. Kludge. It has been the basis for every XBasic menu loader, and has saved us uncounted thousands of hours. The author had asked me not to reveal his identity, but I think I can now tell you that "A. Kludge" was really the late Dr. Stefan Romano, who passed away recently at the age of 57. He was a brilliant man who did much for the TI world, at first as editor of the IUG library, and then through the Amnion library and Amnion Helpline. He was of great help to me on several occasions.

Some of you may have obtained from me a copy of CATWRITER which wrote GOSUB 21 in line 12, and CALL LOADs in lines 21-25 to change the cursor to my Tigercub emblem. If you have begun to have problems with the resulting LOAD program or with my previous Tigercub Menuloader which used the same CALL LOADs, I have finally found out the cause. When my Horizon RamDisk is on, any program containing those CALL LOADs will lock up the second time it is run!



Heartfelt thanks go to Gary Taylor for getting the newsletter out this month while your editor is on vacation. He wears many hats in this group. Hope you all appreciate him.

FROM THE LIBRIAN. . .
Sue Harper

This month let's look at the education section of the library!

There are 114 disks in the EDUC section of the library, ranging from preschool through adult educational information. Also included are some helps for teachers and other educators, such as moms and dads! Many of the programs can be used by the kids without assistance, and thus are perfect for busy families. Some have predetermined information to be covered, such as Let's Build America, a states and capitals program, and some allow input, such as spelling and math programs that ask you to enter the questions or parameters of the quiz.

Some of the varity include:
AERODYII - with lift, gravity, drag and thrust
MATHQUIZ - for teachers to create quizes
MUS-TER-AZ - a quiz about musical terms
SUN - a tutorial about the sun
TEETH - teaches the names of the teeth
XBASIC - tutorials on extended basic
*LOGODUMP - files to use with the LOGO cartridge
SPANISH - translates spanish into english
STURMFACTS- uses Euclid's algorithms
EA/TUTOR - Editor/Asembler tutorials
PROTEINSYN - DNA program
STORM - a program to track hurricanes
*ASTRONOMY - a grand tour of the universe
XYZ-PLOT - generates contour maps
ASPIC - Amature Special Purpose Instructional Code

As for languages, we have Latin, Finnish, Russian, English, Spanish, Vietnamese. . . and who knows what else is to come along!

I hope you can see that the disks in the EDUC section of the library, although we call them school stuff, are not all stuffy! Check them out and see what you can learn!

See you at the meeting. . .

OCT 1991	
S M T W T F S	
6	MEETING
13	
20	
27	BRD. MTG.

THE PUG MEETS
ON THE 2ND SUNDAY OF THE MONTH
AT WHITEHALL BOROUGH COMMUNITY ROOM
100 BOROUGH PARK DRIVE
WHITEHALL, PA.

CLASSES BEGIN AT 3PM
GENERAL MEETING BEGINS PROMPTLY AT 6PM

NOV 1991	
S M T W T F S	
3	MEETING
10	
17	
24	

PUG OFFICERS

Pres:	Gary Taylor	412-341-6874
V Pres:	Rick Keppler	412-941-3559
Treas:	Art Gardner	412-835-4304
Rec Sec:	George Dick	412-793-5834
Librarian:	Susan Harper	412-464-0525
Paper Lib:	Tom Puhatch	412-885-3183
Cor. Sec.:	Gary Taylor	412-341-6874
NL Editor:	Audrey Bucher	412-881-5244

SCHEDULE

3-4PM Questions, problems and answers
4-5PM Disk Plus Aid with Gary TAylor
5-6PM Extended Basic with Mickey Schmitt
6PM-7 General Meeting

DUES $15/YR

PITTSBURGH TI USER'S GROUP
P.O. Box 8043
Pittsburgh, PA 15216

DALLAS TI HC UG

BOX 29863
DALLAS, TX. 75229

DATED MATERIAL
Please Deliver by
Oct 3rd

PUG BBS
412-341-4820
300/1200/2400 BAUD
24 HOURS