THE OFFICIAL NEWSLETTER OF THE TI-99/4A

ISSUE #11 NOVEMBER 1988

FOR THE RECORD

by Ed Bittner
Recording Secretary

The October meeting of the West Penn 99'ers was as crisp as the brisK autumn air. Presidential hopeful ,VP Mickey Schmidt, opened the meeting at 7:10 in Scott Colemans absence. The immediate order of business was the solicitation of nominations from the floor for the upcoming elections. No additional names were added thus the slate remains the same as published last month. With no library report and the recording secretarys report already distributed, corresponding secretary Gene Kelly announced that a "group" rate of $19.95/ unit for TI Base was available. TI Base is a data base similiar to IBM D-BASEII Gene also put in a plug for PUG'S BBS service (824-6779) - TRY IT @!- you'll like it !! (8 BIT, 1 STOP, NO PARITY at 300,1200, 2400 baud).

The discussion centered around the Harrisburg show with Joe Ekl, our other presidential hopeful, speaking at length,(Fortunately, our candidates are not slinging mud, and not running negative ads , nor shoving in our faces POLLS !!@!@!@!

Out-going president Scott Coleman (ala.-dictator-grand -po-ba) announced at a previous meeting that indeed we will have a Pizza Party at the November meeting---so-- don't miss it - What a way to get the vote out..

Demonstration of TI Base by Mike Sealy went well. MacFlix, a TI utility which can use Macintosh files to draw pictures was shown by Gary Taylor, also Gary demonstrated an animated cartoon program of dinosaurs, pretty neat ! Gary promised to bring the Geneve to the Nov, meeting. John Willforth showed with his usual Karisimmaa, Perfect Push and a snake that eats itself, unwillingly, called Nibbler, great action.! Classes in Hardware (J. Willforth) and TI Tips (Tom Mainier) followed the main meeting. The raffle prize was a data base filing program.

Nominations for the Election

Pres.	: M. Schmidt : J. Ekl	V.Pres.	: M. Sealy : S. Coleman
Rec. Sec	: E. Bittner	Treasurer	: J. Trayers
Cores. Sec.	: G. Taylor : G. Kelly	Librarian	: R. Ekl : W. Meyers

Respectfully submitted,
Scoops Bittner

MISC NEWS....

I can't believe the number of hardware articles appearing in newsletters from around the country, and the world. I'd like to print them all, but due to the fact that I'm hearing very little feed back from our own members, who this newsletter is really for, indicating that they are actually constructing the offerings, I don't dare print more than one in any issue. I can imagine how much interest I would have in running to the mailbox for the latest issue of the WEST PENN 99'ER, if it was just chock full of articles on structuring more efficient code for a Fortran Compiler (I don't even know if I said that right).

I would like to tell you about two new ones that we've recieved, that those of you who are interested might not come across unless you do go through newsletter from other groups. If you want a copy of any just write, or call 412 527-6656 address on front of this newsletter.

The first is the POOR MAN'S A to D (I qualify). This article is by John Martin, and appears in Sept. 1988 issue of SNUGLETter from Southern Nevada, and uses a quad opto-isolater chip (PS2501-NEC) to make the joystick port an input port for analog signals which then get digitized by an assembly program he includes which samples stereo music and represents this music on your screen. Apprx. cost $12.- $16.. The system needs E/A, 32K, Disk in order to assemble and run his program, a Mini-mem unit could also make this possible, or a program written in BASIC, XBASIC might be written, but of course would have a problem in the frequency in sampling the input.

The other project is quite interesting, but would be within the capabilities of probably 1 in 10,000 TI users, and this deals with writing your own ROM (operating system) for the TI. The article presents you with instructions on putting one wire, and one diode on your CPU board. That's the easy part, now try writing the changes to be put into EPROM. This is where I believe most of you will stop. If you do tackle this, I'll volunteer to burn the EPROM if my MECHATRONICS will handle the EPOROM you choose. In any case, this project idea comes from CARL VERLAG, of Munich, West Germany. Again if you are interested call or write for your copy of these articles.

I'll have some good hardware at very resonable prices at the November meeting, so bring money. I am helping to distribute an estate. E/A package, TI 32K mem. card, TI RS232/PIO card, Magnavox amber monitor, and many other items of interest.You can contact me earlier if you can't wait, or don't want to lose the opportunity to get something.

FOR SALE.....
1 PASCAL CARD for PEB, all docs and disks, contact John F. Willforth (412) 527-6656 or write. $100.

TREASURER'S REPORT FOR OCTOBER '88

FROM JAN TRAYERS

10/18	CASH ON HAND	$ 40.00
10/18	LIBRARY SALES	4.00
"	MICROPENDIUMS	62.50
	"WRITER'S" BOOK	4.00
	DISK SALES	36.00
	RAFFLE	28.00
	DUES	60.00
	DATA CASES	50.00
	TOTAL	$284.50
	SUPPLIES & RAFFLE PRIZE	- 21.75
		262.75
	POSTAGE	- 67.50
		195.25
10/31	DEPOSIT	- 60.00
	CASH ON HAND	135.25
10/18	BANK BALANCE	$1141.83
10/20	DISK CASES	- 92.28
		1049.55
10/25	MICROPENDIUM	- 30.00
		1019.55
10/31	DEPOSIT	+ 60.00
	BALANCE	1079.55
	TOTAL CASH BALANCE	$1214.80

November 15th
Meeting at 7:00 PM.
- FREE PIZZA PARTY
- ELECTIONS
- DEMONSTRATIONS
- BE THERE OR BE....

GENEALOGY SOFTWARE AVAILABLE FOR THE TI.....

GENEALOGY WORKSHOP.....$49.95.....Micro-Sphere Tenex
14009 E. Jefferson Boulevard
Mishawaka, IN 46545

YOUR ANCESTORS, IN BASIC.....$10.-$20...KEN BARBER
9648 S.E. Ellis
Portland, OR 97266

YOUR FAMILY TREE.....$44.95.....HARDWARE, INC.
P.O. Box 241746
Memphis, TN 38124

"FILL-IN-THE-BLANKS".....$8.00 .....JAN KNAPP 314 428-0752
TI-WRITER GENEALOGY HELPER

JAN KNAPP
2318 ruckert Av.
St Louis, MO 63114

Templates to print out fill-in-the-blank forms with sample filled-in forms. They include: TREE,GROUPSHEET,LARGE GROUPSHEET, ADDRESS PAGE, SOURCE, and CENSUS and a bibliography of sources for those doing genealogical work.

I'M SO EMBARASSED.... another correction on the clock circuit printed in the June '88 issue of the West Penn 99'er and PUG Peripheral. Here are both of the corrections. The chip U2 pin 14 (above pin 8) should be pin 11. This is the pin that outputs to pin 4 of U3. The other correction is that the eight DATA pins that output/input on the right of the clock chip (58167) are all wrong, in that pin 22 of the clock chip (D7) is shown going to pin 34 (D0) of the I/O connector in the Speech Synthesizer, or in the console. It should go to pin 37 (D7) of the I/O connector, and pin 21 (D6) to pin 40 (D6), etc. In other words the data lines are all reversed. Symptoms will be a very odd display of time ([),(#), etc. These are the only corrections (I'm sure). This occured because of my desire to get all the hardware projects I can out to you. I hope that none of you got discouraged because it didn't work. For USERS GROUPS who either handed out this project or reprinted it, I and your members will sure appreciate it if you publish these corrections. J.F. W.

PASCAL/p-CODE PART 12
Stan Katzman

I would like to discuss some more about files in the U.C.S.D. Pascal system.

In standard Pascal you cannot have random file access, but in U.C.S.D. Pascal you can. Random access is done using the Seek statement. The Seek syntax is "Seek(Filename,Recnum)". In the Seek mode you can both read and write data to the disk. In the Seek statement the "Filename" variable is the name of the file and the "Recnum" variable is the record number. The record numbering starts at zero. The illustration of the Seek statement is given in the rather long program included.

There are several conditions that must be observed when Seek is used: 1)The file must be of type "File", 2)when you "Seek" a record the Seek statement must be followed by a "Get" or a "Put" statement (see program) and 3)the syntax for displaying a field on the screen or working with one field is done thusly (as shown in the program) "Readln(Order^.Name)" or "Writeln(Order^.Name)" or you can use the "With" statement such as "With Order^ Do".

Some miscellaneous thoughts about Pascal files: Reading any Pascal file from disk is slower than BASIC; if you save numbers as reals it will slow up the reads so it will take minutes in order to read a disk file. So if possible do not store data as reals. Lastly if you save a file with the ".TEXT" suffix you will be able to read the file in the Editor. You can also modify a ".TEXT" file in the Editor.

DISK DRIVES (#3)
by John F. Willforth

Last month you received the basic schematic of a disk drive tester. This month, I'll describe the functions and give you a schematic for a power supply to drive the unit and the disk drive under test.

The large connector on the left (J1) is the ribbon cable that goes to the drive's logic board. The small connector to the right of center near the top (J2) is the power cable to the drive. Rotary switch (S1) is the unit select switch which will select the drive by the strapping you have set on the drive. MOTOR ON (S2) turns on the drive motor, makes it easier to test this associated circuitry in the drive, the DIRECTION of head stepping (S3), in or out, WRITE GATE control (S4), mode selector for the drive, WRITE DATA signal (S5) to the write circuitry in the drive logic, STEP in the DIRECTION selected (S6),provide write data for the WRITE DATA line (S7)when the WRITE GATE is enabled, and do all this on or to the SIDE selected (S8).

You can watch to see if you are getting INDEX pulses on D1, and if DATA read from the drive is present on D2, or see when the heads are at TRACK 00 on D3, and if the WRITE PROTECT sensor is working at D4.

You can further check the drive speed at TP2 (Scope or frequency counter), or look at the signal coming off the read head at TP2 (scope or null meter), if you are on an alignment track for disk alignment.

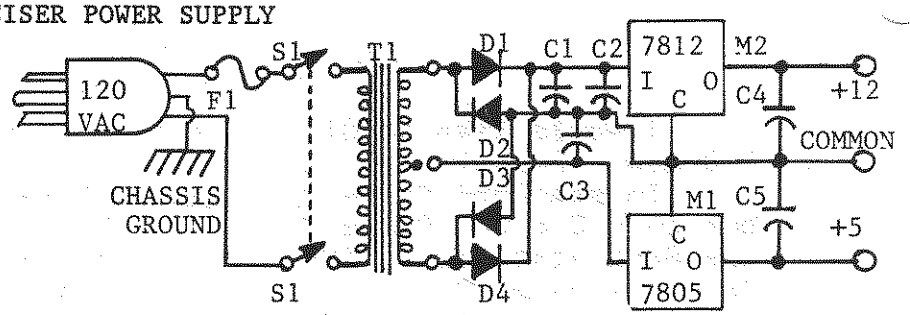
This unit has a lot of uses (for an unintelligent device) and enables easy benchtesting and circuit tracing will be much easier.

The power supply shown below must be built at your own risk. Dangerous high voltage exists, and only experienced electronics people should construct this part of the Disk Exerciser project. You could use a Triple Output Powersupply available from Radio Shack for sometime now, 277-1022 provided you were only testing 1/2 height disk drives, or you could also use an old TI console power supply in the same way, and hook up the appropriate pins to the three points indicated in the upper right hand corner of the schematic of the exerciser.

EXERCISER POWER SUPPLY

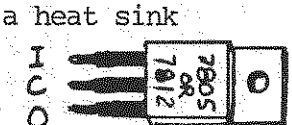
PARTS:

1 Fuse Holder w/1A fuse.. F1
1 Switch DPST 115 VAC.... S1
1 Transformer 18VAC C.T
  Radio Shack 273-1515... T1
4 3A diodes 1N5402
  R.S. 276-1143..D1,D2,D3,D4
3 Capacitors 2200MFD 35V.
  R.S. 272-1020.....C1,C2,C3
2 Capacitors 100MFD 35V.
  R.S. 272-1016.....C4,C5
1 +5 Volt Regulator
  R.S. 276-1771.....M1
1 +12 Volt Regulator
  R.S. 276-1770.....M2
1 AC Power cable



Observe polarity on any capacitor marked. "+" toward +5 and +12. It is also IMPERATIVE to attach the two regulators to large HEAT SINK.

Be sure to use a heat sink compound and firmly mount the 2 regulator components to a large metal mass.



The power supply above will also make a very good source of DC power for a stand-alone disk drive, as long as the unit does not draw over one amp. on the +12 volt line. This unit will get very hot due to the very high (18 VAC) on the secondary. The reg.s have to drop this to 12, and the +5 v. reg. must also work very hard because it is dropping 9 VAC to the needed +5. If a transformer that outputs about 16 VAC c.t., can be acquired, the unit will run cooler. After you have constructed this unit, and put it into the box with the EXERCISER, connect the three lines +12, COMM.(GROUND), and +5 to these pts. in the EXERCISER. We'll begin next month with a disk drive.

This series could be a lot of fun, but will also a lot of work. John F. Willforth Nov. 1988

EZ-KEYS, A REVIEW

by Bill Gaskill - Topics - LA 99ers, CA - Apr '88

EZ-KEYS is a new product from Asgard Software, designed with the Extended Basic programmer or Extended Basic program user in mind. To use it one must have Extended Basic, 32K memory expansion, and a disk drive. The program retails for $14.95 and is currently available directly from Asgard Software or TENEX Computer Express. I am certain it will be available from other major 99/4A retailers in the near future.

A MACRO GENERATOR

EZ-KEYS is one of three keyboard generators I know of available for the 99/4A owner. PC-KEYS II, from Techni-Graphics, and SoftKeys from Quality 99 Software, are the other two. The fact that all three programs could be generically grouped into the "macro" development classification is really about all that they have in common. PC-KEYS II offers disk cataloging, a pop-up notepad, a pop-up calendar, and user-definable/selectable screen dump capability, along with the ability to define a limited number of "hot-keys" that perform common functions with one keystroke. SoftKeys is basically a "hot-key" macro generator without all of the added features of PC-KEYS II or the additional cost.

EZ-KEYS takes a different approach to the concept of macro generation. It too allows you to define "hot-keys", but the keys you define can do much, much more than either of the other two programs. In fact, by my definition, EZ-KEYS is really the only true macro generator of the three. In my experience as a user of "other" computers, macros are short programs that "remember" keystrokes for you, so you can later call them up at the press of a single key. In other words, they are time savers that shorten the number of steps you have to go through to perform a desired function or series of functions.

On all of the MS-DOS macro generators I have used, macro files are built in one of two ways. One method is to use a "remember" mode that tracks and then saves keystrokes as you press them and another method is to provide a macro editor that allows you to write and save small files containing the desired keystrokes. EZ-KEYS is of the second type. Instead it provides a macro editor that appears at the press of a single key.

EZ-KEYS allows up to 55 keys to be defined for macro use, with each macro capable of being 668 characters (about 7/8 of a screen) in length. Perhaps the neatest feature of EZ-KEYS is its ability to link macros together. This means one macro file can RUN another, thus providing almost unlimited potential to the utility the EZ-KEYS environment can offer the XB programmer or user. EZ-KEYS also RUNs Extended Basic programs or parts of XB programs.

For instance, if you wished to have a disk cataloging program available at the touch of a key, you could write it in Extended Basic, LIST it to disk so it is SAVEd in DV/80 format, then define a macro for it. In fact, such a program is provided on the EZ-KEYS disk.

Programming a macro is simple if you are creating simple macros. It can become quite complex and demanding if you really want to build some sophisticated applications. Saving a macro is simple and straightforward. You simply define it in the Macro Editor, exit out of the editor, do a BYE at the READY prompt and then load the program again. You will immediately be given the option of loading or saving your macros. When you save them, all macros that exist in memory are saved to disk for use any time the EZ-KEYS environment is loaded.

To use your macros in a RUNning program environment you simply edit a line in the EZ-KEYS program so it RUNs the first program you want to activate, then you must save the EZ-KEYS program as LOAD. When the EZ-KEYS LOAD program is read into memory it brings all of your macro definitions with it, then RUNs your first program. That's it! No programming expertise required here, just a user-friendly common sense approach to interfacing with your XB program(s).

Extended Basic programs that use assembly language subroutines may also be used with EZ-KEYS. The author has included an EZLOADER that will allow you to save custom routines and your macros all together. Assembly routines are loaded first, then your macros so they do not conflict with the subroutines already in low-memory. The whole package is then saved as a memory-image file and can be called up whenever you use the application with the custom assembly routines. The really neat thing here is the ability to customize the EZ-KEYS environment to fit as many different uses or programs as you have.

A PROGRAMMING UTILITY

EZ-KEYS is an assembly language coded program, designed to operate in an Extended Basic environment. Aside from its ability to generate macros it also provides a set of utilities for the Extended Basic programmer. While in the command mode (at the] READY prompt) in XB EX-KEYS allows you to set a timer that will automatically SAVE your work in case of a power failure or interruption. The time intervals can be from 0 to 18 minutes apart and two files, BACKUP1 and BACKUP2 are used to save your work. All worked is saved only to DSK1. Another option allows the setting of background and foreground colors in the programming environment, much the same as the GrafKracker Utilities or John Johnson's Horizon RAM Disk menu allow. Colors may also be set for the Macro Editor and the special characters displayed in a macro file.

Another routine will highlight numbers and arithmetic operators so they appear on screen in the reverse color of the background and foreground colors chosen. When a running program is being used you may also set all character sets to the same color by linking to a routine named RCOLOR.

Although the manual cautions Extended Basic might not always be able to interpret it, EZ-KEYS lets you write a

single program line that can be 23 screen lines long. How's that for expanding the capabilities of Extended Basic? Additionally, you can press Function 7 or Function 6 to move the cursor directly between program lines while in the programming (immediate) mode.

You may also write macro files in the TIW Editor, in a manner similar to writing a .BAT file for the GENEVE or any MS-DOS machine. The author included a customized CHARAI file to use on your TI-Writer disk. This file contains the character definitions needed to display the special characters that represent specific macro functions. This is another example of the forethought put into this program. I would guess few first-time users would opt for this method of development though, since it requires the use of the Transliterate Mode in TI-Writer and is only sparsely documented in the EZ-KEYS manual. There is a chart in the documentation that shows the various equivalents that are available.

Once you have created the macro file you simply print it to disk, run the POKER program provided on the EZ-KEYS disk, and the macro file is then assigned as a macro definition.

If EZ-KEYS has a shortcoming it is in the method used to call macros from a RUNning XB program. The cursor must be displayed on screen for a macro to be accessed. You cannot, for instance, access a macro when the program being RUN is looping at a CALL KEY statement. Once the file instructions within a macro have been set in motion they are suspended only by a "hold" command in the macro (a Ctrl H). So you must have programmed a Control H in the macro file so it appears at the proper point in your XB program. This can be tricky and a little confusing to the novice programmer. I would have rather seen an execution routine used that could be called at any time.

One curious oversight in the EZ-KEYS program involves character definitions. A custom character set is used in the macro generator and is not reset when an XB program is used. You end up with a couple of lower case letters out of line with the standard TI chracter set in your running XB program. While this is easily overcome by restoring the offending characters with CALL CHAR statements in the program you are running, it would be nice to see EZ-KEYS take care of this for you. It is one less than professional aspect of a program that is otherwise truly representative of "commercial" quality software.

EASE OF USE

While creating macros is not in the suggested domain for a new programmer, setting up the EZ-KEYS program to use macros is. More detailed documentation is needed to help the less adventuresome wade through the rigors of complex macro development.

DOCUMENTATION

The documentation is adequate for simple macro definitions, but falls short of being a complete tutorial for the advanced macro programmer. It covers most of the "absolutes" of macro development in the EZ-KEYS environment, but stops short of really being a useful guide to the advanced macro developer. However, in fairness to Asgard Software, it's difficult to offer such an outstanding product and couple it with outstanding documentation for $14.95. The documentation is well written and understandable, an important consideration in any new software purchase. If EZ-KEYS "takes off" perhaps Asgard will develop follow-up products for it such as a disk of predefined macros or a tutorial on advanced macro programming.

VALUE

Harry Wilhelm is the author of EZ-KEYS. I know nothing about Mr. Wilhelm nor do I recall ever reading his name in any of the many 99/4A publications. After seeing the product he has produced, I hope he continues to write programs for the 99/4A (and hopefully the 9640). If future Wilhelm applications are anything like EZ-KEYS, we are all in for a treat. EZ-KEYS is a superb first-release application. It is well thought out, professionally executed and virtually error free. For the adventuresome programmer or user EZ-KEYS promises unlimited potential and utility. It is truly a professional application that needs only more complete documentation and some fine tuning to push it into the "stellar" software class. If you don't have EZ-KEYS you should buy it. You won't regret the meager $14.95 investment. Even if you do not use it to do ANY macro development of your own, you will likely find another, perhaps more important, use for it. I would not be surprised to see future XB type applications developed under the EZ-KEYS environment. It is truly a powerful development tool that cries out for an imaginative programmer to come along and demonstrate some of its potential. With the right combination of good marketing, dependable customer support and continued development of the product, EZ-KEYS could become a standard among 99/4A users. It is THAT GOOD! It is only in its infancy in version 1.0. I am sure the best is yet to come.

FINAL GRADE

You will note EZ-KEYS falls down to a "B" rating in some categories. In the PERFORMANCE area I knocked it down because of the less than flexible method used to call up a macro and the program's inability to suspend macro operations more effectively. In the EASE OF USE area I took some points away because of the complexities required to design more than just simple macros. The DOCUMENTATION lost points because of brevity and several typos that snuck into the final product. The VALUE category makes up for all of the little shortcomings I found in this first release. It is the "missing link" that we have been looking for in making more of the 99/4A than just a single purpose machine. For $14.95 you simply can't go wrong. If the TI Community supports EZ-KEYS like it deserves, I am confident the incentive will be present for Harry Wilhelm to continue development of the product. I am equally certain other talented programmers will develop applications to run in the EZ-KEYS environment. The end result to our support of this product is sure to be an even better product in the future.

REPORT CARD:

Performance.....B
Ease Of Use.....B
Documentation....B
Value.....AAA
FINAL GRADE.....A

Extended BASIC Speaking Program LISTer
by Chris Schram

There are those out there in the Texas Orphanage who truly seem to possess "The Gift" for inventing new ways of utilizing that funny little monster. Then there are the rest of us who can from time to time, if we're lucky, assemble the little bits and pieces of others' genius into something, if not completely original, at least suitable to our own needs. This article is about a program I would not have been able to write had others not first paved the way.

I was intrigued by Steven Richardson's proofreader program from the June 1987 issue of MICROpendium, but the fact that it would only operate in console BASIC and required the TE-II module made it unsuitable for me. Most of the programs I enter are in Extended BASIC. I also found the "LIST a screenful, then RUN" nature of Mr. Richardson's program a bit awkward, to say the least.

I set out to put the pieces together. I already owned a copy of TEXT-TO-SPEECH (ENGLISH) (PHD 5076), so getting Extended BASIC to talk would be no problem. I also had a copy of Barry Traver's TOKEN/READ, a program that PEEKs into Expansion RAM and displays the words associated with the Extended BASIC tokens it finds. Most of my program was lifted from Mr. Traver's program. Mr. Traver gives credit to John Clulow and Michael Riccio for inspiration. I must do the same, for they, indirectly, assisted me, too.

The following two programs were originally one stand-alone program that could be MERGEd into what you wanted LISTed. The only problem was that the TEXT-TO-SPEECH machine code ate up so much memory that it was really only able to list itself. Not very useful. The way it stands now, you RUN the LISTINIT program just once to create a disk file that is used by the LIST program. That saves a heap of memory at the sacrifice of some speed because of all the disk access.

Note: If you own the TEXT-TO-SPEECH program, then you already know if it works with your hardware setup. I know that it does NOT work with the FOUNDATION expansion memory cards and there may be a few surprises with other configurations. In other words, first determine if it's worth it to proceed.

Keeping everything on DSK1, for the time being, type in the two programs below. (Be careful, the LIST program uses @, [,], _, and \ as variable names.) Copy SPEAK, XLAT, SETUP, and DATABASE from the TEXT-TO-SPEECH disk. Run LISTINIT to create the LISTDATA file which contains the text and speech strings used by LIST. OLD the program you want to list. MERGE the LIST program. RUN. You will be given a chance to select what lines you want to list. You can watch the tokens as they appear one by one on the screen while the program speaks. The screen listing closely resembles a normal LISTing. Press [Fctn 4] when you've seen/heard enough. If you want to RUN the program under test, just REM out line 2.

```
1 ! SAVE DSK1.LISTINIT
100 !!!!!!!!!!!!!!!!!!!!!!!!
110 ! EX. BASIC SPEAKING!
120 !   PROGRAM LISTER    !
130 !  ****************   !
140 !  * INITIALIZER *    !
150 !  ****************   !
160 !  by: Chris Schram   !
170 !    San Jose, CA     !
180 !     July 1987       !
190 !!!!!!!!!!!!!!!!!!!!!!!!
200 ! Requires:
210 ! TI EXTENDED BASIC
220 ! SPEECH SYNTHESIZER
230 ! EXPANSION MEMORY
240 ! DISK MEMORY
250 ! TEXT-TO-SPEECH
      (ENGLISH) DISKETTE
      (PHD 5076)
260 !
270 ! Note:
280 ! I have found that
      the TEXT-TO-SPEECH
      program does not
      work with the
      FOUNDATION expansion
290 ! memory card. There
      may be other
      hardware/software
      conflicts yet to be
      discovered.
300 DIM C$(1,255)
310 ON ERROR 320 :: X$="INIT
IALIZING...." :: DISPLAY AT(
12,7)ERASE ALL:X$ :: CALL LI
NK("XLAT","^)"&X$,B$):: CALL
LINK("SPEAK",B$,43,128):: G
OTO 350
320 CALL INIT :: CALL LOAD("
DSK1.SPEAK","DSK1.XLAT","DSK
1.SETUP")
330 CALL LINK("SETUP","DSK1.
DATABASE")
340 RETURN
350 ON ERROR STOP :: FOR I=3
2 TO 127 :: C$(0,I)=CHR$(I):
: NEXT I :: FOR I=129 TO 254
:: READ C$(0,I):: NEXT I ::
C$(0,34)=CHR$(34)&CHR$(34)
360 READ I,C$(1,I):: IF I<25
5 THEN 360
370 OPEN #1:"DSK1.LISTDATA",
OUTPUT,DISPLAY ,FIXED 26,REL
ATIVE
380 FOR I=0 TO 255
390 IF C$(1,I)="" THEN X$=C$
(0,I)ELSE X$=C$(1,I)
400 CALL LINK("XLAT","^"&X$,
B$):: PRINT #1,REC I:CHR$(LE
N(C$(0,I)))&C$(0,I);TAB(12);
SEG$(B$&RPT$(CHR$(0),15),1,1
5)
410 DISPLAY AT(23,1):C$(0,I)
;TAB(12);B$:C$(1,I):: CALL L
INK("SPEAK",B$,43,128)
420 NEXT I
430 CLOSE #1 :: CALL CLEAR :
: STOP
440 DATA ELSE,::,!,IF,GO,GOT
O,GOSUB,RETURN,DEF,DIM,END,F
OR,LET,BREAK,UNBREAK,TRACE,U
NTRACE,INPUT,DATA,RESTORE,RA
NDOMIZE
450 DATA NEXT,READ,STOP,DELE
TE,REM,ON,PRINT,CALL,OPTION,
OPEN,CLOSE,SUB,DISPLAY,IMAGE
,ACCEPT,ERROR,WARNING,SUBEXI
T,SUBEND,RUN
460 DATA LINPUT,,,,,,,THEN,TO
,STEP,",",;,:,),(,&,,OR,AND,
XOR,NOT,=,<,>,+,-,*,/,^,,,,,
EOF,ABS,ATN,COS,EXP,INT
470 DATA LOG,SGN,SIN,SQR,TAN
,LEN,CHR$,RND,SEG$,POS,VAL,S
TR$,ASC,PI,REC,MAX,MIN,RPT$,
,,,,,,,NUMERIC,DIGIT,UALPHA,S
IZE,ALL
480 DATA USING,BEEP,ERASE,AT
,BASE,,VARIABLE,RELATIVE,INT
ERNAL,SEQUENTIAL,OUTPUT,UPDA
TE,APPEND,FIXED,PERMANENT,TA
B,#,VALIDATE
490 DATA 0,END OF LINE,32,SP
ACE,33,EX CLUHMATION,34,QUOT
E QUOTE,35,POUND SIGN,38,AMP
ERSAND,39,APOSTRUHFEE,43,PLU
S,44,COMMA
500 DATA 45,DASH,46,DOT,58,C
OLEN,59,SEMEECOLEN,60,LESS T
HAN,62,GREATER THAN,63,QUEST
ION MARK,91,( BRACKET,92,REV
ERSE SLANT
510 DATA 93,) BRACKET,94,CIR
CUMFLEX,95,UNDER LINE,96,GRA
VE,123,LEFT BRACE,124,VERTIC
AL LINE,125,RIGHT BRACE,126,
TILDUH,127,DEL
520 DATA 130,DOUBLE COLEN,13
1,TAIL REM,134,GO 2,135,GO S
UB,142,BRAKE,143,UNBRAKE,147
,DAYTUH,162,>DISPLAY,165,AIR
OR
530 DATA 167,SUB X IT,168,SU
B END,177,2,179,COMMA,180,SE
MEECOLEN,181,COLEN,184,AMPER
SAND,188,X OR,191,LESS THAN
540 DATA 192,GREATER THAN,19
3,PLUS,194,MYNUS,195,TIMES,1
96,DIVIDED BY,197,RAISED TO
THE POWER,202,E O F,203,A B
S,204,A T N
550 DATA 205,CO SINE,206,E X
P,207,I N T,209,S G N,210,S
INE,211,S Q R,212,TAN GENT,2
14,C H R $,215,RAND,217,P O
S
560 DATA 219,S T R $,220,A S
C,222,REC ERD,225,R P T $,2
34,U AL FUH,239,E RACE,243,V
AREEUHBL,244,REL UH TIV,253,
NUMBER
570 DATA 249,>APPEND,251,PER
MANENT,199,QUOTE,201,LINE,2
55,>INITIALIZING
```

CONT. ON PAGE 8

```
1 ! SAVE DSK1.LIST,MERGE
2 CALL LIST :: STOP :: !@P-
32000 !@P+
32002 SUB LIST
32004 @=0 :: [=1 :: ]=2 :: _
=12 :: \=256 :: GOTO 32008 :
: L$,Q$,X$,Y$ :: A,A1,A2,B,C
,D,F,I,J,J1,J2,K,L,QF,S,T,UF
:: CALL KEY :: CALL PEEK ::
DIM Y(3)
32006 !@P-
32008 OPEN #[:"DSK1.LISTDATA
",INPUT ,DISPLAY ,FIXED 26,R
ELATIVE
32010 LINPUT #[,REC 255:X$ :
: DISPLAY AT(_,7)ERASE ALL:"
INITIALIZING...." :: CALL SP
EAK(SEG$(X$,_,15))
32012 LINPUT #[,REC 199:Q$ :
: Q$=SEG$(Q$,_,15)
32014 LINPUT #[,REC 201:L$ :
: L$=SEG$(L$,_,15)
32016 ON ERROR STOP :: CALL
FIND(3,F):: CALL PEEK(F,A,B)
:: F=A*\+B :: CALL FIND(3200
0,T):: CALL PEEK(T+4,A,B)::
T=A*\+B
32018 DISPLAY AT(_,5)ERASE A
LL:"LIST FROM?";F :: ACCEPT
AT(_,16)VALIDATE(DIGIT)SIZE(
-5)BEEP:F
32020 CALL FIND(F,J1):: CALL
PEEK(J1,A,B):: F=A*\+B :: D
ISPLAY AT(_,16)SIZE(5):STR$(
F)
32022 DISPLAY AT(14,_):"TO?"
;MAX(F,T):: ACCEPT AT(14,16)
VALIDATE(DIGIT)SIZE(-5)BEEP:
T
32024 CALL FIND(T,J2):: CALL
PEEK(J2,A,B):: T=A*\+B :: D
ISPLAY AT(14,16)SIZE(5):STR$
(T)
32026 DISPLAY AT(18,[):"* PR
ESS ANY KEY TO PAUSE *"
32028 FOR L=J1 TO J2 STEP -4
32030 CALL PEEK(L,A,B,C,D)::
J=C*\+D-65537 :: CALL PEEK(
J,A1)
32032 X$=STR$(A*\+B):: DISPL
AY X$&" ";:: CALL SPEAK(L$):
: CALL LNUM(X$)
32034 FOR I=[ TO A1 :: S=0 :
: CALL PEEK(J+I,Y(0),Y(1),Y(
2),Y(3))
32036 IF Y(S)=199 THEN QF=Y(
S+[)+[ :: I=I+] :: S=S+] ::
UF=é :: DISPLAY CHR$(34);::
CALL SPEAK(Q$):: IF QF=[ THE
N DISPLAY CHR$(34);:: CALL S
PEAK(Q$)
32038 IF Y(S)=200 THEN UF=Y(
S+[)+[ :: I=I+] :: S=S+] ::
QF=é
32040 IF Y(S)=201 THEN X$=ST
R$(Y(S+[)*\+Y(S+])):: Y$=""
:: I=I+] :: S=S+] :: GOTO 32
044 ELSE LINPUT #[,REC Y(S):
X$
32042 Y$=SEG$(X$,_,15):: X$=
SEG$(X$,),ASC(SEG$(X$,[,[))
32044 DISPLAY X$;:: IF Y$=""
THEN CALL LNUM(X$)ELSE CALL
SPEAK(Y$)
32046 LINPUT #[,REC Y(S+[):Y
$ :: K=ASC(SEG$(Y$,[,[))
32048 IF (LEN(X$)>[ OR K>[)A
ND(QF<>] AND UF<>])THEN DISP
LAY " ";
32050 QF=QF-[ :: UF=UF-[
32052 IF QF=[ THEN DISPLAY C
HR$(34);:: CALL SPEAK(Q$)
32054 IF UF=[ AND K>[ THEN D
ISPLAY CHR$(32);
32056 CALL KEY(@,K,S):: IF S
<[ THEN 32062
32058 DISPLAY BEEP:;
32060 DISPLAY AT([,[)SIZE(28
):"* PRESS ANY KEY TO RESUME
*" :: DISPLAY AT([,[)SIZE(2
0): :: CALL KEY(@,K,S):: IF
S<[ THEN 32060
32062 NEXT I
32064 DISPLAY
32066 NEXT L
32068 DISPLAY BEEP: : :"PRES
S ANY KEY TO CONTINUE"
32070 CALL KEY(@,k,S):: IF S
=@ THEN 32070
32072 GOTO 32018
32074 !@P+
32076 SUBEND
32078 SUB FIND(L,Y)
32080 \=256 :: GOTO 32082 ::
A,B,C,D,L1,X :: CALL PEEK :
: !@P-
32082 CALL PEEK(-31952,A,B,C
,D):: X=A*\+B-65536 :: Y=C*\
+D-65539
32084 IF X>=Y THEN SUBEXIT
32086 CALL PEEK(Y,A,B):: L1=
A*\+B :: IF L1<L THEN Y=Y-4
:: GOTO 32084
32088 !@P+
32090 SUBEND
32092 SUB SPEAK(Q$)
32094 GOTO 32096 :: CALL INI
T :: CALL LINK :: CALL LOAD
:: !@P-
32096 ON ERROR 32098 :: CALL
LINK("SPEAK",B$,43,128):: S
UBEXIT
32098 CALL INIT :: CALL LOAD
("DSK1.SPEAK","DSK1.SETUP")
32100 CALL LINK("SETUP","DSK
1.DATABASE")
32102 RETURN
32104 !@P+
32106 SUBEND
32108 SUB LNUM(X$)
32110 [=1 :: _=12 :: GOTO 32
112 :: Y$ :: I :: !@P-
32112 FOR I=[ TO LEN(X$)
32114 LINPUT #[,REC ASC(SEG$
(X$,I,[)):Y$ :: Y$=SEG$(Y$,_
,15):: CALL SPEAK(Y$)
32116 NEXT I
32118 !@P+
32120 SUBEND
```

GLOSARY OF COMPUTER TERMS

BIT: Describes computers, as in "OUR" computer cost quite a BIT.
BOOT: What your friends do to you when you brag about your computer.
BUG: What your eyes do after staring at a screen too long.
CHIPS: Used to insert into DIP while working at your computer.
COPY: What you do at school cause you were playing PACMAN so much last night.
CUSOR: What you become when your computer breaks down.
DISK: What slips in your back after hours of sitting down.
DUMP: Where all your hobbies go after buying a computer.
ERROR: Made when you walked into the computer store "just to look".
EXPANSION UNIT: The room you add to your house to store your computer.
FLOPPY: The condition of the user's muscles after sitting around and eating chips.
HARDWARE: Rakes, moweres, and other things you haven't touched this summer.
MENU: What you'll never seee again, cause now you're too poor to eat out.
PROGRAM: What you used to watch on the TV, until you hooked the couputer to it.
RAM: What you do to the side of your of your computer when it's broken.
RETURN: What you do with the computer after RAM doesn't work.
WINDOW: What you throw the computer thur when you con't RETURN IT.

HELP FOR "PUSH" USERS WITH RAM DISKS.....by John F. Willforth

There is a very intricate game called PUSH, I have not yet been able to find out for sure it's reimbursement requirements, (commercial, freeware, public domain) but it is a very challenging game. The problem I have is that it takes awhile to load from a disk drive, and so using a string search utility I sought the ASCII strings that dealt with the disk designations "DSK1.", which allowed it to load it's two support programs from the same disk that the D/F program "PUSH" resided on, DSK1.. This was not as easy as it would first appear. The string "DSK1." was not found! I then took a sector editor, and slowly stepped through the sectors of the "PUSH" file, and found two very close strings that fortunately did modify to the desired drive. This is just to save you a little time in case you desire to do the same with this program.

In the ASCII mode search for "DSBK1F" and change the "1" to the drive number of the RAM DISK you want the program to run from, and then search for "DSBK1B" and change the "1" to the same drive as the first change made.

In HEX mode search for "4453424B3146" and change "31" (1) to "34" (4) drive 1 to 4, and then search for "4453424B3142" and change "31" (1) to "34" (4) drive 1 to 4. If any other drive, 2, 3, 5, 6, 7, etc. use "32", "33", "35", "36", "37", etc respectively. Thats all there is to it! Note that the search will be easier and certainly more reliable if the only file on the disk at the time is "PUSH", but I found it with many on the disk because none that were on the disk had these particular strings present.

```
100 ! ORACLE
110 ! VERSION XB.2.1
120 ! 08 MAR 85
130 ! BY JIM SWEDLOW
140 !
150 DISPLAY AT(10,4)ERASE AL
L BEEP:"** I AM  THE ORACLE
**" :: CALL DELAY :: RANDOMI
ZE
160 CALL INIT :: CALL PEEK(-
28672,I):: IF I=0 THEN DISPL
AY AT(20,1):"I cannot operat
e without the    Speech Synt
hesizer!" :: STOP
170 DISPLAY AT(15,1):"  I a
nswer all questions": : :"As
k qustions with  YES or NOan
swers -- When you are donepr
ess ENTER."
180 CALL DELAY :: DISPLAY AT
(24,1):"  PRESS ANY KEY TO
BEGIN"
190 CALL KEY(0,I,S):: IF S=0
 THEN 190 ELSE CALL CLEAR
200 PRINT : :"WHAT IS YOUR Q
UESTION?" :: INPUT Q$ :: IF
Q$="" THEN 220
210 CALL DELAY :: CALL REPLY
(Q$):: CALL DELAY :: GOTO 20
0
220 DISPLAY AT(10,1)ERASE AL
L:"THANK YOU FOR CONSULTING"
: : : : : : :"    ** THE ORA
CLE **" :: CALL DELAY :: STO
P
230 !
240 SUB DELAY :: FOR I=1 TO
200 :: NEXT I :: SUBEND
250 !
260 SUB REPLY(A$)
270 A$=SEG$(A$,1,2):: IF A$=
"WH" OR A$="HO" THEN CALL OT
HER ELSE CALL YESNO
280 SUBEND
290 !
300 SUB YESNO
310 ON INT(10*RND)+1 GOTO 32
0,330,320,340,350,350,360,37
0,380,390
320 CALL SAY("YES"):: SUBEXI
T
330 CALL SAY("I THINK SO"):
 SUBEXIT
340 CALL SAY("LOOKS POSITIVE
"):: SUBEXIT
350 CALL OTHER :: SUBEXIT
360 CALL SAY("LOOKS NEGATIVE
"):: SUBEXIT
370 CALL SAY("I DO NOT THINK
 SO"):: SUBEXIT
380 CALL SAY("NO WAY"):: SUB
EXIT
390 CALL SAY("NO"):: SUBEND
400 !
410 SUB OTHER
420 ON INT(10*RND)+1 GOTO 43
0,440,450,460,470,480,490,50
0,510,520
430 CALL SAY("I CAN NOT TELL
 YOU THAT"):: SUBEXIT
440 CALL SAY("SAY THAT A DIF
FERENT WAY"):: SUBEXIT
450 CALL SAY("YOU DO NOT WAN
T TO KNOW"):: SUBEXIT
460 CALL SAY("I DO NOT KNOW"
):: SUBEXIT
470 CALL SAY("I AM NOT SUPPO
SED TO SAY"):: SUBEXIT
480 CALL SAY("I WILL NOT TEL
L YOU"):: SUBEXIT
490 CALL SAY("I CAN ONLY GUE
SS"):: SUBEXIT
500 CALL SAY("I CAN NOT ANSW
ER THAT"):: SUBEXIT
510 CALL SAY("I DO NOT REMEM
BER"):: SUBEXIT
520 CALL SAY("TRY SOME THING
 ELSE"):: SUBEND
```

NEW SOFTWARE IS RELEASED ..... NUTMEG TI-99ERS...OCT., 1988

From Asgard Software, P.O. Box 10306, Rockville, MD 20850

1. COLUMN ATTACK: Fast action arcade game written in Fortran 99. Defend Earth against rampaging alien spaceships. Requires 32K, disk system and Extended Basic. $9.95 + $.75 S/H.
2. DINOSAURS: TI-Artist Graphics. 2 disks of fun for all ages. Has dinosaur font, background scenery & dinosaur animation. Create pictures, stickers, cards, reports. $12.95 +$.75 S/H.
3. QUICK-RUN: Extended basic utility makes other XB programs run instantly. Can eliminate time consuming program initialization. Takes "snap-shot" of program & saves to disk. $9.95 + $.75 S/H.

Cont. page 11

From Genial Computerware, P.O. Box 183, Grafton, MA 01519

1. MacFLIX: Allows TI users to view, print and save graphics produced by MacIntosh MacPaint. Can be saved in TI Artist _P format. $15.00 + $1.00 S/H.
2. GRAPHICS EXPANDER: Version 2.0 reduces & expands TI-Artist fonts & instances. Also CSGD fonts & small graphics. Will also convert formats between these two programs. Many other features. $10.00 + $1.00 S/H.
   Owners of Version 1.0 may send original disk & $3.00.

TWO JOYSTICKS IN ONE.....BY Curtis Borders.....C.O.N.N.I.E.

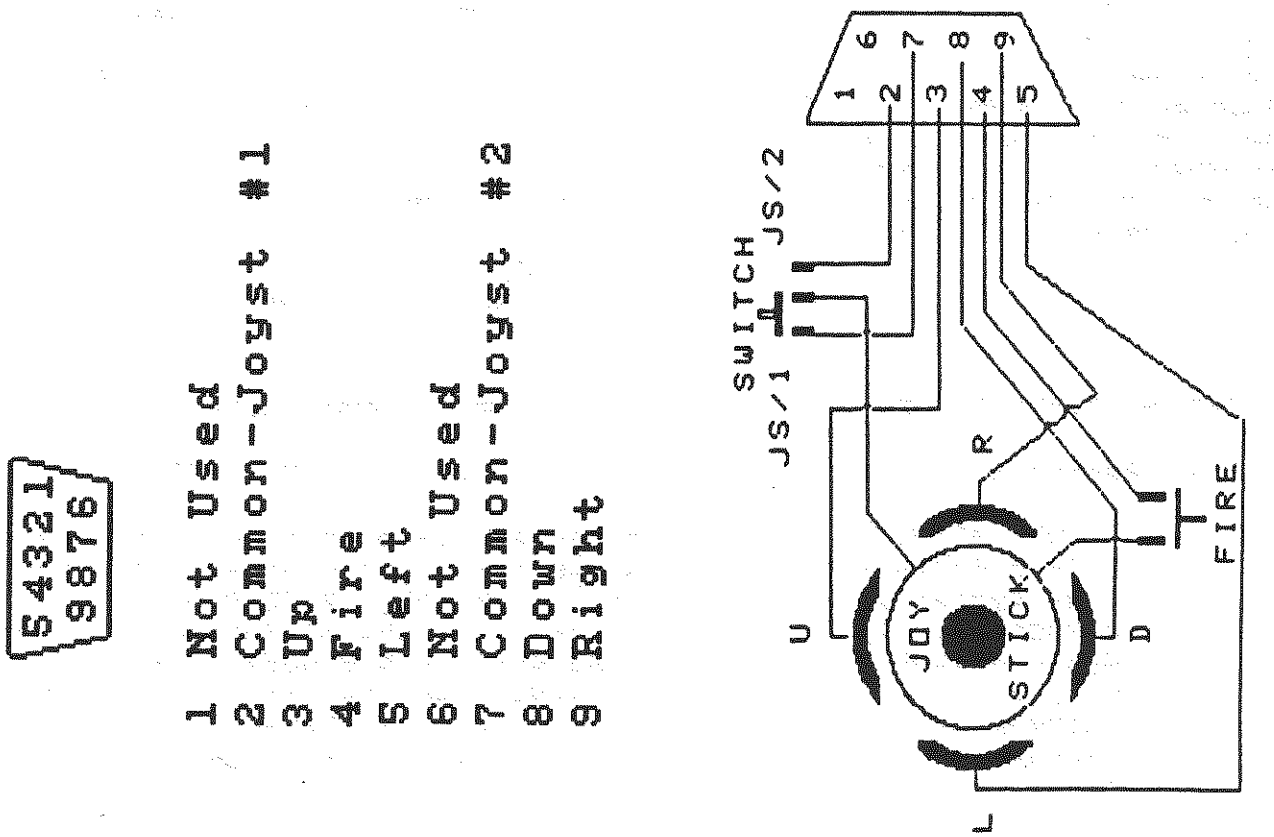
This is how I made two joysticks out of one:
First you will need one of those surplus joystick cables. All the pins will have to be there with the exception of pin 1 and 6. (Pins 1 and 6 are not used on the TI 99/4A) You can get one at "Star Surplus" on N. High St. Columbus, OH. They sell for about $1.99.

OK, now that we have the cable, take your favorite joystick- it can be Atari, Boss, EPSX500XJ, or Wico, but I wouldn't waste my time on TI joysticks. Take the joystick apart and unsolder the cable from the connectors, all but the ground (or common) wire. That's the wire that connects all the pads together.

Take your new cable and an ohm meter and write down all the pin numbers and what color wire it is, because all color codes may not be the same. Take the _____ color wire from pin 2 to one of the outside terminals of the switch, then take the _____ color wire from pin 7 to the other side of the switch.

Take a short piece of wire from the center of the switch and solder it to the ground (common) wire. If the switch is wired up right, when it is in one position, you will be using joystick number 1, and when it is in the other position, you will be using joystick number 2.

Take the _____ color wire from pin 3 to the joystick up position.
Take the _____ color wire from pin 8 to the joystick down position.
Take the _____ color wire from pin 9 to the joystick right position.
Take the _____ color wire from pin 5 to the joystick left position.
Take the _____ color wire from pin 4 to the joystick fire button. The other side of the fire button will go to the ground (or common) wire.
Use schematic below.



Slot Machine

FROM PUNN, NOVEMBER 1988

Planning a trip to Reno or Las Vegas? If so you will want to try this program and practice up for your trip.

You've probably been there in the past at one time or another. So you will recall the the whirring and the sounds of the slot machines and found yourself wondering what the fascination was.

You"ll soon find out when this program turns your computer into a fabulous Nevada style fruit machine. All the playing instructions you need will appear on the screen. At certain points you will be presented with a list of options.

When you see:

Insert, Hold, Play OR End

enter your choice by typing the first letter of the option you want, for instance P keeps your machine playing. The reels are numbered 1, 2, and 3. If you want to hold one or more reels, type in the appropriate number or numbers after you have entered H for Hold.

Lights will flash and music play as the wheels whiz around. Have you won this time?

Keep playing and sooner or later you are bound to hit the Jackpot.

```
10 REM TI(EXTENDED)-SLOT MACHINE
20 CALL CLEAR :: RANDOMIZE
30 DIM WF(13),R(2,13),JAC(13):: JAC(5)=-1 :: JAC(6)=-1 :: JAC(7)=-1
40 FOR VV=1 TO 13 :: READ WF(VV),R(1,VV),R(2,VV):: NEXT VV
50 DATA 110,0,0,116,0,0,115,0,0,109,0,0,106,8,50,105,8,40,114,7,30
60 DATA 113,5,25,104,3,20,136,3,20,137,2,10,128,2,10,112,0,0
70 DIM JPX(4),JPY(4),JF(4),JCC(4):: FOR JO=1 TO 4 :: READ JPX(JO),JPY(JO),JF(JO):: NEXT JO
80 DATA 11,8,110,12,8,116,14,8,115,15,8,109
90 NJ$=CHR$(138)
100 DIM WPX(3),WPY(3),V(4),HVV(3):: WPX(1)=11 :: WPX(2)=13 :: WPX(3)=15 :: FOR I=1 TO 3 :: WPY(I)=10 :: NEXT I
110 DIM HPX(3),HPY(3),HF$(3):: FOR I=1 TO 3 :: HPX(I)=WPX(I):: HPY(I)=13 :: NEXT I
120 H1$=CHR$(107):: HH$=CHR$(108):: HH$=CHR$(96):: HO$=H1$&HH$&H1$&HH$&H1$
130 DIM GPX(4),GPY(4),GV(4):: GPX(1)=22 :: GPY(1)=1
140 GPX(2)=27 :: GPY(2)=2 :: GPX(3)=22 :: GPY(3)=3 :: GPX(4)=17 :: GPY(4)=2
150 WJ=4 :: WH=0 :: WG=0 :: GW=0 :: WI=0 :: WT=-1
160 GOTO 1740
170 REM BLINK S1$/S2$ AND GET IN
180 DISPLAY AT(PLY,PLX):S1$;:: CH=0 :: FOR DE=1 TO 10
190 IF CH=0 THEN CALL KEY(0,GET,CH)
200 NEXT DE :: DISPLAY AT(PLY,PLX):S2$;:: IF CH=1 THEN RETURN
210 FOR DE=1 TO 10 :: IF CH=0 THEN CALL KEY(0,GET,CH)
220 NEXT DE :: IF CH=1 THEN RETURN
230 GOTO 180
240 REM AND INCREMENT TO MONEY
250 FOR AD=SGN(IN)TO IN STEP SGN(IN)
260 DISPLAY AT(1,7):MO+AD;
270 IF SGN(IN)>0 THEN 290
280 CALL SOUND(-50,200,2):: GOTO 300
290 FOR SO=500 TO 700 STEP 100 :: CALL SOUND(-50,SO,2):: NEXT SO
300 NEXT AD :: MO=MO+IN :: RETURN
310 REM REMOVE DOUBLE
320 DO=0 :: CALL HCHAR(2,3,32,7)
330 FOR JO=1 TO 4 :: CALL HCHAR(JPY(JO),JPX(JO)+2,JF(JO)):: JCC(JO)=0 :: NEXT JO
340 WJ=4 :: RETURN
350 CALL HCHAR(24,3,32,28):: RETURN
360 REM JACKPOT
370 FOR TI=1 TO 4 :: CALL HCHAR(4,3,32,JC)
380 FOR C=1 TO JC :: CALL SOUND(-100,150+20*C,0):: DISPLAY AT(4,C):"J";:: NEXT C
390 NEXT TI :: RETURN
400 REM DEFINE CHARACTERS
410 CALL CHAR(96,"FFFFFFFFFFFFFFFF")
420 CH$="0F0F0F0F0F0F0F0F" :: CALL CHAR(97,CH$)
430 CALL CHAR(60,CH$)
440 CALL COLOR(9,6,1,10,2,16,11,7,16,12,16,6,13,11,16,14,13,16)
450 CALL CHAR(107,"FFBBBBBBB8BBBBBB")
460 CALL CHAR(108,"FFFFFFFFFFFFFFFF")
470 A$="1B98FF3D3C3CE404"
480 CALL CHAR(109,A$):: CALL CHAR(115,A$)
490 A$="1B19FFBC3C3C2720"
500 CALL CHAR(110,A$):: CALL CHAR(116,A$)
510 CALL CHAR(136,"02043C566A566A3C")
520 CALL CHAR(114,"00006C7C7C381000")
530 CALL CHAR(112,"0204387C7C7C3800")
540 CALL CHAR(104,"02020C3C7B7B3000")
550 CALL CHAR(128,"0040606070301E00")
560 CALL CHAR(137,"020413C37C7C3800")
570 CALL CHARPAT(36,A$):: CALL CHAR(106,A$)
580 CALL CHARPAT(63,A$):: CALL CHAR(120,A$)
590 CALL CHAR(113,"081C3E7F3EE1C0800")
600 CALL CHAR(104,"081C2A772A080800")
610 CALL CHAR(105,"00103877C7C103800")
620 CALL CHAR(138,"0000000000000000")
630 RETURN
640 REM$$DRAW SCREEN
650 CALL CLEAR :: CALL SCREEN(12)
660 CALL HCHAR(7,12,96,7)
670 FOR I=8 TO 12 :: CALL HCHAR(I,11,96,9):: NEXT I
680 CALL VCHAR(7,20,60)
690 CALL VCHAR(8,20,97,4)
700 CALL HCHAR(12,20,96)
710 CALL HCHAR(13,12,96,7)
720 FOR I=14 TO 19 :: CALL HCHAR(I,13,96,5):: NEXT I
730 CALL HCHAR(20,12,96,7)
740 CALL HCHAR(21,11,96,9)
750 CALL HCHAR(22,10,96,11)
760 DISPLAY AT(1,1):"TOTAL:0"
770 DISPLAY AT(10,3):CHR$(112);" = 1";
780 FOR VV=12 TO 5 STEP -1
790 DISPLAY AT(23-VV,1):"-";RPT$(CHR$(WF(VV)),2);"-=";R(1,VV);
800 DISPLAY AT(23-VV,20):RPT$(CHR$(WF(VV)),3);"=";R(2,VV);
810 IF JAC(VV)THEN DISPLAY AT(23-VV,27):"*J*";
820 NEXT VV :: RETURN
830 REM $$ ADAPT VARIABLES
840 WT=WT+1 :: IF WI>0 THEN HB=-1 :: WI=0
850 FOR HO=1 TO 3 :: HF$(HO)=NH$ :: DISPLAY AT(HPY(HO),HPX(HO)):NH$;:: NEXT HO
860 IF DO THEN 910
870 FOR JO=1 TO 4 :: IF JCC(JO)=0 THEN 900
880 JCC(JO)=JCC(JO)-1 :: IF JCC(JO)>0 THEN 900
890 WJ=WJ+1 :: CALL HCHAR(JPY(JO),JPX(JO)+2,JF(JO))
900 NEXT JO :: GOTO 920
910 DC=DC-1 :: IF DC=0 THEN GOSUB 320
920 IF JA THEN DISPLAY AT(4,JC):" ";:: JC=JC-1 :: IF JC=0 THEN JA=0
930 RETURN
940 REM $$ NOT ENOUGH MONEY
950 HB=-1 :: GOSUB 320
960 JA=0 :: CALL HCHAR(4,3,32,15)
970 GOSUB 350 :: DISPLAY AT(24,1):"INSERT OR END";
980 S1$="INSERT (#)" :: S2$=RPT$(" ",10):: PLX=1 :: PLY=2 :: GOSUB 180
990 IF GET=ASC("P")THEN 980 ELSE RETURN
1000 REM $$ HOLD POSSIBLE
1010 DISPLAY AT(24,1):"INSERT, HOLD,PLAY OR END";
1020 S1$=HO$ :: S2$=HF$(1)&HH$&HF$(2)&HH$&HF$(3)
1030 PLX=11 :: PLY=13 :: GOSUB 180
1040 IF GET<49 OR GET>51 THEN RETURN ELSE HO=GET-48
1050 NH=NH+1 :: IF HF$(HO)=NH$ THEN HF$(HO)=H1$ ELSE HF$(HO)=NH$
1060 GOTO 1020
1070 REM $$ NO HOLD
1080 GOSUB 350 :: DISPLAY AT(24,1):"INSERT,PLAY OR END";
1090 S1$=RPT$(CHR$(120),3):: S2$=RPT$(CHR$(96),3):: PLX=12 :: PLY=19 :: GOSUB 180
1100 RETURN
1110 REM $$ WHAT TO DO WITH WINNINGS?
1120 GOSUB 350 :: IF NOT HB THEN DISPLAY AT(24,1):"HOLD, ";
1130 DISPLAY AT(24,7):"GAMBLE OR COLLECT";
1140 GOTO 1090
1150 REM $$ SPIN GAMBLE WHEELS
1160 FOR I=7 TO 10 :: CALL HCHAR(I,20,32):: CALL HCHAR(I+1,20,60):: NEXT I
1170 CALL SOUND(-4000,-7,29)
1180 FOR I=10 TO 7 STEP -1 :: CALL HCHAR(I,20,60):: CALL HCHAR(I+1,20,97):: NEXT I
1190 FOR WD=1 TO 3 :: IF HF$(WD)=NH$ THEN DISPLAY AT(WPY(WD),WPX(WD)):CHR$(138);
1200 NEXT WD
1210 FOR WD=1 TO 3 :: IF HF$(WD)=H1$ THEN 1240
1220 FI=INT(RND*100+1):: IF FI<5 THEN IF JCC(FI)<>0 THEN V(WD)=13 ELSE V(WD)=FI :: JCC(FI)=-1 :: GOTO 1240
1230 V(WD)=5-(FI>7)-(FI>10)-(FI>13)-(FI>23)-(FI>36)-(FI>49)-(FI>68)-(FI>87)
1240 FOR DE=2 TO 300 :: NEXT DE :: DISPLAY AT(WPY(WD),WPX(WD)):CHR$(138);
1250 CALL SOUND(-100,300,2):: CALL HCHAR(WPY(WD),WPX(WD)+2,WF(V(WD)))
1260 CALL SOUND(4000,-7,29):: NEXT WD :: CALL SOUND(-1,-2,30):: RETURN
1270 REM TAKE CARE OF JOKERS
1280 J=0 :: FOR WD=1 TO 3 :: VV=V(WD)
1290 IF VV>4 THEN FV=VV :: GOTO 1320
1300 JW=WD :: J=J+1 :: IF JCC(VV)>0 THEN 1320
1310 DISPLAY AT(JPY(VV),JPX(VV)):NJ$;:: JCC(VV)=20 :: WJ=WJ+1
1320 NEXT WD :: DO=(WJ=0)::
IF DO THEN DC=15 :: DISPLAY AT(2,1):"DOUBLE!";
1330 RETURN
1340 REM $$ COMPUTE WINNINGS
1350 HV=13 :: ON J+1 GOTO 1360,1380,1360,1410
1360 IF NOT(JA AND FV>10)THEN HV=FV
1370 GOTO 1410
1380 V(0)=V(3):: V(4)=V(1):: IF JA AND V(JW-1)<>V(JW+1)THEN 1410
1390 V(0)=15 :: V(4)=15
1400 IF V(JW+1)>V(JW-1)THEN HV=V(JW-1)ELSE HV=V(JW+1)
1410 FOR WD=1 TO 3 :: IF V(WD)<5 THEN HVV(WD)=HV ELSE HVV(WD)=V(WD)
1420 NEXT WD :: IF HVV(1)<>HVV(2)OR HVV(2)<>HVV(3)THEN 1440
1430 IF JAC(HVV(1))THEN JA=-1 :: JC=15 :: GOSUB 370
1440 FOR WD=1 TO 3 :: IF HVV(WD)=13 THEN WI=WI+1
1450 NEXT WD :: IF JA THEN WI=10*WI
1460 WS=-(HVV(1)=HVV(2))-(HVV(2)=HVV(3)):: IF WS>0 THEN WI=WI+R(WS,HVV(2))
1470 IF DO THEN WI=2*WI
1480 RETURN
1490 REM $$$ GAMBLE ROUTINE
1500 DT=1 :: GV(1)=2*WI :: GV(2)=0 :: GV(3)=INT(3*WI/2):: GV(4)=INT(WI/2)
1510 KEY=0 :: WG=WG+1 :: GOSUB 350 :: DISPLAY AT(24,1):"STOP";
1520 FOR LI=1 TO 3 :: CALL HCHAR(LI,16,96,17):: NEXT LI
1530 RR=RR+1 :: IF RR>4 THEN RR=1
1540 DISPLAY AT(GPY(RR),GPX(RR)):STR$(GV(RR));
1550 CALL SOUND(-4000,150+50*RR,2)
1560 IF KEY<>ASC("S")THEN CALL KEY(0,KEY,CH):: GOTO 1580
1570 DT=(1+RND*2)*DT :: FOR DE=1 TO DT :: NEXT DE :: IF DT>150 THEN 1590
1580 CALL HCHAR(GPY(RR),GPX(RR)+2,96,4):: GOTO 1530
1590 FOR LI=1 TO 3 :: CALL HCHAR(LI,16,32,17):: NEXT LI
1600 GW=GW+GV(RR)-WI :: WI=GV(RR):: CALL SOUND(-1,150+50*RR,2):: RETURN
1610 REM $$$ END OF GAME
1620 CALL CLEAR :: CALL CHARSET :: CALL SCREEN(8)
1630 DISPLAY AT(5,1):"AMOUNT OF MONEY:"
1640 [MAGE " $ ####.##"
1650 DISPLAY AT(7,3):"PUT IN:";:: DISPLAY AT(7,18):USING 1640:MI
1660 DISPLAY AT(8,3):"GOT BACK:";:: DISPLAY AT(8,18):USING 1640:MO/4
1670 DISPLAY AT(9,3):"MAX AT ONE TIME:";:: DISPLAY AT(9,18):USING 1640:MM-4
1680 DISPLAY AT(10,3):"WON BY GAMBLING:";:: DISPLAY AT(10,18):USING 1640:GW/4
1690 DISPLAY AT(12,1):"NUMBER OF HOLDS: ";NH
1700 DISPLAY AT(13,1):"NUMBER OF GAMBLES:";WG
1710 DISPLAY AT(14,1):"NUMBER OF TURNS: ";WT
1720 RETURN
1730 REM MAIN PROGRAM
1740 GOSUB 410 !DEF CHAR
1750 GOSUB 650 !SCREEN
1760 GOSUB 840 !ADAPT
1770 ON -2*HB-(MO>1)+1 GOSUB 950,1010,950,1080
1780 DISPLAY AT(5,9):: CA=-(GET=ASC("I"))-2*(GET=ASC("P"))-3*(GET=ASC("E"))
1790 ON CA+1 GOTO 1770,1800,1820,2020
1800 WI=WI+1 :: IN=4 :: GOSUB 250
1810 GOTO 1770
1820 IN=-2 :: GOSUB 250
1830 GOSUB 1160 !SPIN WHEELS
1840 IF DO THEN 1860
1850 GOSUB 1280 !JOKERS
1860 GOSUB 1350 !WINNINGS
1870 IF HF$(1)=H1$ OR HF$(2)=H1$ OR HF$(3)=H1$ THEN 1890
1880 HB=0 :: LW=WI :: IF WI>0 THEN 1910 ELSE 1760
1890 HB=-1 :: IF WI>LW THEN 1910
1900 DISPLAY AT(5,9):" YOU LOST" :: CALL SOUND(-600,200,2):: GOTO 1760
1910 DISPLAY AT(5,9):"YOU WON";:: DISPLAY AT(5,17):WI;:: FOR DU=1 TO WI :: CALL SOUND(-50,500,1):: CALL SOUND(1,500,30):: NEXT DU
1920 IF MO<2 THEN HB=-1
1930 GOSUB 1120 !GET INSTRUCTION
1940 CA=-(GET=ASC("H")AND NOT HB)-2*(GET=ASC("G"))-3*(GET=ASC("C"))
1950 ON CA+1 GOTO 1930,2010,1990,1960
1960 IN=WI :: GOSUB 250
1970 IF WI>MM THEN MM=WI
1980 GOTO 1760
1990 HB=-1 :: GOSUB 1500 !GAMBLE
2000 IF WI>0 THEN 1910 ELSE 1900
2010 WI=0 :: GOTO 1760
2020 GOSUB 1620 !END
2030 END
```

WEST PENN99ERS
% John F. Willforth
R. D. # 1 BOX 73A
JEANNETTE, PA
15644

NOVEMBER 1988 ISSUE

NEXT MEETING TUESDAY NOV. 15th
7:00 PM at the UNITED PRESBYTERIAN CHURCH OF THE COVENANT 4th and Oak
IRWIN, PA

TEXAS INSTRUMENTS HOME COMPUTER
USERS GROUP

MICKEY SCHMITT
196 BROADWAY AVE.
LOWER BURRELL PA 15068